ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2015
Т. 4, № 1

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Учредитель — Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Южно-Уральский государственный университет» (национальный исследовательский университет)

Тематика журнала охватывает следующие основные нарпавления (не ограничиваясь ими):

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Математическое и программное обеспечение высокопроизводительных вычислительных систем
- Системное программирование
- Распределенные вычисления, облачные и грид-технологии
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY

2015
Vol. 4, no. 1

SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

ISSN 2305-9052

Vestnik Yuzhno-Ural'skogo Gosudarstvennogo Universiteta.
Seriya "Vychislitel'naya Matematika i Informatika"

South Ural State University

The scope of the journal includes, but not limited to, the following topics:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- High performance computer software
- System programming
- Distributed, cloud and grid computing
- Programming technology
- Computer graphics
- Internet technologies
- E-learning
- Database and knowledge processing
- Data mining

Содержание

Информатика, вычислительная техника и управление

Вычислительная математика

Contents

УДК 004.94 DOI: 10.14529/cmse150101

ОБЗОР СИСТЕМ ПРОЦЕДУРНОЙ ГЕНЕРАЦИИ ИГР

М.Г. Меженин

Процедурная генерация контента (ПГК) является одной из наиболее актуальных задач в индустрии видеоигр. Под ПГК понимают автоматическое создание различных составляющих частей игр; особый интерес представляет проблема автоматического создания игровых правил и целых игр. В статье представлен обзор исследований, посвященных данной проблеме; описываются алгоритмы генерации игровых правил и игр различных жанров, в том числе алгоритмы на основе парадигм эволюционного моделирования и логического программирования, а также способы автоматической оценки генерируемых игр. Кратко рассмотрены первые системы генерации игр, реализованные в универсальных игровых программах. Также описаны различные форматы представления и специализированные языки для описания игровых правил в подходящем для алгоритмической обработки виде.

Ключевые слова: процедурная генерация контента, игровой дизайн, универсальные игровые программы, искусственный интеллект.

Введение

Процедурная генерация контента является одним из наиболее актуальных и активно развивающихся направлений исследований в сфере мультимедиа, в частности в индустрии видеоигр. Под *процедурной генерацией контента (ПГК)* понимают автоматическое и полуавтоматическое создание и динамическое изменение различных составляющих частей игр, в том числе игровых объектов и уровней, двумерной и трехмерной графики, эффектов, звуков, музыки, персонажей, сюжетов и др. [10]. Использование ПГК позволяет не только значительно понизить стоимость создания контента, но и решает проблему персонализации, приобретающую большую значимость в связи с увеличением количества потенциальных игроков.

В исследованиях алгоритмов ПГК особенный интерес представляет проблема процедурной генерации игровых правил, которые лежат в основе всех, в том числе и неэлектронных, игр. Решение данной задачи обладает практической значимостью, поскольку процедурная генерация правил позволит не только существенно разнообразить видеоигры за счет динамического изменения правил в процессе игры (в том числе, адаптируя игровой процесс под желания игрока), но и, возможно, создать совершенно новые игровые механики и жанры [24]. Кроме того, системы генерации игровых правил могут быть полезны исследователям алгоритмов искусственного интеллекта в сфере универсальных игровых программ, а также игровым дизайнерам для быстрого создания и тестирования различных игровых прототипов [15].

Процедурная генерация игр невозможна без языка описания игровых правил, то есть некоторого строго-определенного формата их представления в подходящем для алгоритмической обработки виде. Разработка такого языка описания обладает теоретической значимостью, поскольку ее результатом будет строгий научно-терминологический аппарат, который позволит формализовать и описать пространство возможных игровых правил [24, 25].

Процедурная генерация игровых правил является достаточно новой областью исследований. В данной статье рассмотрены ключевые работы в этой сфере, посвященные разработке систем генерации игр и языков их описания.

Статья организована следующим образом. В первом разделе рассмотрены исследования в области универсальных игровых программ. Во втором разделе описаны современные системы процедурной генерации видеоигр. В третьем разделе кратко рассмотрены различные языки описания игр. В заключении суммируются сделанные на основе данного обзора выводы и рассматриваются направления дальнейших исследований.

1. Универсальные игровые программы

В конце 1980-х годов одной из основных и наиболее актуальных задач в области искусственного интеллекта являлась задача создания компьютерной программы для игры в шахматы. С каждым годом шахматные программы становились всё умнее, и уже в 1989 году компьютерная система Deep Thought выиграла в демонстрационном матче у международного мастера по шахматам Дэвида Леви. Вместе с тем, данные системы были настолько сильно специализированы на игру в шахматы, что, по мнению некоторых ученых, достижения в данной области уже не могли использоваться для решения более общих задач искусственного интеллекта (ИИ) [24].

Для решения данной проблемы была предложена концепция *универсальной игровой программы* (УИП), то есть такой системы ИИ, которая была бы способна играть в различные игры, не обладая какими-либо знаниями о конкретной игре до ее начала. Одной из первых работ в данной области является система *METAGAME* [17], состоявшая из двух основных подсистем: модуля ИИ, способного играть в различные настольные игры, и *генератора игр*, назначение которого заключалось в генерации игр для тестирования алгоритмов ИИ.

Таким образом, METAGAME генерирует игры из многочисленного, но конечного множества «симметричных шахмато-подобных игр», удовлетворяющих следующим основным критериям:

— игры происходят на прямоугольной доске с квадратными клетками;
— в игре принимает участие двое игроков;
— начальные позиции фигур и правила симметричны для обоих игроков.

Множество возможных игр в METAGAME задается с помощью формальной *игровой грамматики*, которая определяет начальное расположение фигур, правила их движения и взятия, условия победы и др. Пример описания конкретной игры в формате METAGAME приведен на рис. 1.

Конкретные игры генерируются с помощью случайной выборки из данной грамматики, без проверки на их проходимость, поскольку в общем случае данная проблема является NP-полной. Для отсечения тривиальных игр, авторы реализовали несколько простых проверок, например, не допускающих игры, цель которых заключается в достижении фигурами своей начальной позиции.

Некоторые из параметров генератора также могут задаваться пользователем перед запуском системы. В частности, такими параметрами являются:

— сложность правил (максимальная длина описания правил движения и взятия для каждой фигуры);

— сложность предоставляемого игроку выбора (например, при генерации более простой версии шахмат игроку может не предоставляться выбор, на какую фигуру заменяется пешка при достижении последней горизонтали);
— глубина поиска (например, при меньшей глубине поиска могут рассматриваться игры с меньшим размером доски);
— локальность (максимальное расстояние, преодолеваемое фигурой за один ход).

```
DEFINE man
 MOVING
  MOVEMENT
   LEAP
   <1,1> SYMMETRY {side}
  END MOVEMENT
 END MOVING
 CAPTURING
  CAPTURE
   BY {hop}
   TYPE [{opponent} any_piece]
   EFFECT remove
                                    MOVEMENT
                                     HOP BEFORE [X = 0]
                                         OVER [X = 1]
                                         AFTER [X = 0]
                                     HOP OVER [{opponent} any_piece]
                                     <1,1> SYMMETRY {side}
                                    END MOVEMENT
                                   END CAPTURE
                                  END CAPTURING
                                  PROMOTING
                                   PROMOTE TO king
                                  END PROMOTING
                                  CONSTRAINTS continue_captures
                                 END DEFINE
```

Рис. 1. Частичное описание игры «американские шашки» на языке METAGAME

Несмотря на то, что процедурная генерация игровых правил не была основной целью проекта METAGAME, эта работа оказала большое влияние на последующие исследования в данной области. Многие последующие языки описания и системы генерации игр во многом основываются именно на результатах METAGAME.

Так, в проекте *Gala* [12] было предложено развитие концепций METAGAME с целью создания универсальной игровой программы для игр с неполной информацией. В рамках проекта был предложен одноименный декларативный язык описания игр на основе языка Prolog и алгоритмы поиска оптимальных стратегий для игр, определенных с помощью этого языка.

```
game(blind_tic_tac_toe,
 [players : [a, b],
  objects : [grid_board : array('$size', '$size')],
  params : [size],
  flow : (take_turns(mark,unless(full),until(win))),
  mark : (choose('$player', (X, Y, Mark),
            (empty(X, Y), member(Mark, [x, o]))),
          reveal('$opponent', (X, Y)),
          place((X, Y), Mark)),
  full : (\+(empty(_, _)) ->
  outcome(draw)),
  win : (straight_line(_, _, length = 3, contains(Mark)) ->
           outcome(wins('$player')))]).
```

Рис. 2. Описание игры «слепые крестики-нолики» на языке Gala

В отличие от языка METAGAME, с помощью языка Gala можно описать на порядок большее множество игр, а именно настольные игры на прямоугольной доске с полной и неполной информацией для одного, двух и более игроков. По мнению авторов, их язык описания игр более лаконичен и прост для понимания. На рис. 2 приведен пример

описания на языке Gala игры «слепые крестики-нолики», отличающейся от обычной версии тем, что во время своего хода игрок сообщает лишь клетку, на которую он сходил, не указывая, поставил ли он в нее «крестик» или «нолик».

Язык Gala позволяет задавать некоторые параметры игры в виде переменных; такими параметрами могут быть в том числе и правила движения фигур. Поскольку генерация игровых правил не была целью проекта Gala, его авторы предлагают простой программный интерфейс, благодаря которому для генерации правил на языке Gala можно использовать систему METAGAME.

2. Процедурная генерация игр

2.1. Генерация настольных игр

Исследования в области универсальных игровых программ были в первую очередь ориентированы на изучение алгоритмов ИИ, и генерация игр в них либо не рассматривалась вовсе, либо была основана на простейших алгоритмах.

В работе В. Хома и Д. Маркса [11] задача автоматического создания игр впервые была рассмотрена как отдельная актуальная проблема в области ИИ. Целью работы было создание алгоритма для генерации *сбалансированных* игр, то есть игр, в которых все игроки находятся в равных условиях и имеют одинаковые шансы выиграть. Отметим, что симметричность игры не гарантирует ее сбалансированность, поскольку даже при идентичности правил для обоих игроков игрок, ходящий первым, может обладать преимуществами над вторым игроком (и, в редких случаях, наоборот).

Авторами работы был предложен термин *автоматический игровой дизайн*, под которым понималась автоматическая корректировка баланса игры путем изменения ее правил. Отметим, что идея динамического изменения игрового баланса рассматривалась и в более ранних исследованиях, однако в них корректировка баланса достигалась путем оптимизации заранее определенных параметров игры, а не самих игровых правил.

Работа Хома и Маркса основывалась на коммерчески успешной универсальной игровой программе Zillions of Games (Zillions Development Corp., 1998), игры для которой описывались пользователями вручную на специальном декларативном языке ZRF. Система Zillions of Games предназначалась для визуализации любых настольных игр на доске, описанных на языке ZRF, с возможностью игр в них с УИП.

Предложенный алгоритм автоматического изменения баланса основан на генетическом алгоритме и заключается в следующем. На каждой итерации алгоритма из некоторого набора игр путем изменения и комбинирования их правил генерируются новые игры, в каждую из которых УИП Zillions of Games играет сама с собой 100 матчей. Каждая игра оценивается с помощью функции оценки по шкале баланса (большую оценку получают игры с минимальной разницей между количеством побед у двух игроков) и шкале разнообразия (большую оценку получают игры, наименее похожие на ранее сгенерированные результаты), после чего формируется новый набор из старых и новых игр с наибольшей оценкой и начинается следующая итерация.

Использование генетических алгоритмов и оценка результатов некоторой симуляции, а не самих особей, являются достаточно распространенными методами процедурной генерации контента; в данной работе было продемонстрировано, что эти методы могут быть успешно использованы и для динамической генерации игр.

В дальнейшем данные методы были развиты в проекте *Ludi* [1], одним из главных достижений которого является игра Yavalath — первая в мире коммерчески изданная игра, полностью сгенерированная программой в автоматическом режиме.

В рамках проекта Ludi был разработан специальный функциональный язык для описания настольных игр на доске, названный Ludi GDL. Описания на данном языке состоят из абстракций высокого уровня, что позволяет легко и лаконично описывать различные игры. Ludi GDL также является расширяемым, но для успешной работы любые дополнительные абстракции необходимо реализовывать и на низком уровне. Пример описания игры «крестики-нолики» на языке Ludi GDL приведен на рис. 3.

```
(game Tic-Tac-Toe
  (players White Black)
  (board (tiling square i-nbors)(shape square)(size 3 3))
  (end (Allwin(in-a-row 3)))
)
```

Рис. 3. Описание игры «крестики-нолики» на языке Ludi GDL

При оценивании игр в системе Ludi акцент делается на поиск наиболее интересных игр. Для этой цели по каждой сгенерированной игре в Ludi проводится некоторое количество автоматических матчей с двумя компьютерными игроками, после чего игра оценивается по 57 критериям, разделенным на три основные группы:

— *объективные* критерии для оценки качеств самих правил независимо от тестовых матчей;
— критерии *играбельности (playability)* для оценки результатов тестовых матчей;
— *качественные* критерии для оценки динамики тестовых матчей.

Авторами отмечается, что объективные критерии были наименее эффективны в поиске интересных игр. Критерии играбельности, характеризующие интересность и базовые свойства игрового процесса, в Ludi повторяют описанные выше критерии сбалансированности игр с добавлением критерия средней длины матча.

Особый интерес представляют качественные критерии, основанные на анализе *историй лидирования* в тестовых матчах, то есть данных о степени лидирования одного игрока над другим в каждый из ходов. Пример истории лидирования приведен на рис. 4.

Рис. 4. Пример истории лидирования

В данном примере перед своей победой на 27-м ходу белый игрок находится в проигрышной позиции с 18 по 22 ход. По мнению авторов, если при некотором наборе правил во многих матчах в процессе игры происходит смена лидера, данная игра является более «драматичной» и интересной.

Генерация игр в Ludi реализована на основе стандартных методов генетического программирования. В рамках тестирования системы с помощью Ludi было сгенерировано 1389 игр, из которых 19 были признаны авторами достаточно интересными, а еще 2, названные Yavalath и Pentalath, — крайне интересными. Описание игры Yavalath на языке Ludi GDL приведено на рис. 5. Правила Yavalath схожи с «крестиками-ноликами» на гексагональной доске; главное отличие — игрок побеждает, поставив четыре фигуры в ряд, но проигрывает, поставив в ряд только три фигуры.

```
(game Yavalath
  (players White Black)
  (board (tiling hex)(shape hex)(size 5))
  (end (Allwin(in-a-row 4)) (Alllose (in-a-row 3)))
)
```

Рис. 5. Описание игры Yavalath на языке Ludi GDL

Дополнительное правило может показаться излишним, но на самом деле оно добавляет игре стратегическую глубину. Например, на рис. 6 приведена позиция, в которой ход #1 белого игрока вынуждает черного игрока сделать блокирующий ход #2, и тем самым проиграть из-за нового правила. Таким образом, игроки могут в некоторой степени управлять ходами противника с помощью нетривиальных последовательностей ходов – подобное появление сложных стратегий из достаточно простых правил подтверждает высокое качество и интересность данной игры.

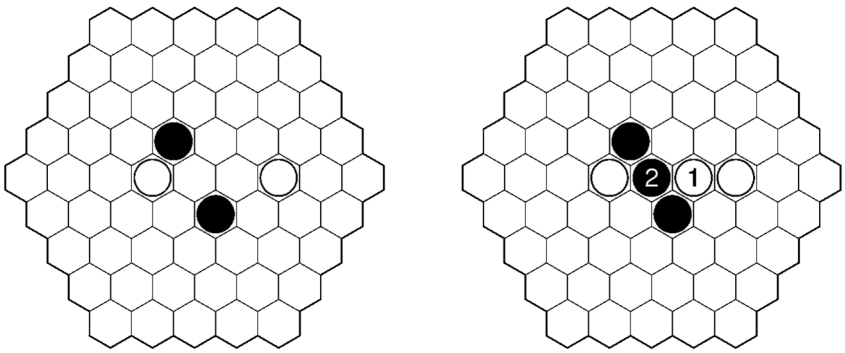


Рис. 6. Выигрышная последовательность ходов белых в Yavalath

Yavalath и Pentalath были впоследствии изданы в коммерческом виде издательством Nestorgames и являются одними из наиболее успешных игр в их каталоге.

2.2. Генерация аркадных видеоигр

В конце 2000-х годов несколько исследователей параллельно начали работу над системами процедурной генерации правил для аркадных видеоигр. Под аркадными играми мы будем понимать двумерные и трехмерные игры, игровой процесс которых заключается в движении некоторых объектов, их столкновении, появлении и исчезновении, и так далее. Данный набор действий характеризует достаточно базовые понятия, из кото-

рых, однако, состоит множество классических игр, таких как Pac-Man (Namco, 1980), Super Mario Bros. (Nintendo, 1985) и Space Invaders (Taito, 1978).

В отличие от шахмато-подобных игр, в аркадных видеоиграх игрок обычно принимает роль некоторого персонажа, движениями которого он управляет в реальном времени. Другими распространенными элементами данного жанра являются управляемые компьютером противники, изменяющие характеристики игрока бонусы, и карта-уровень с преградами. Эти и другие отличия аркадных видеоигр от настольных представляют особый интерес в исследовании алгоритмов их процедурной генерации.

Один из первых опытов в данном направлении описан в работе Дж. Тогелиуса [22], в которой предложен способ генерации простых двумерных игр, подобных знаменитой игре Pac-Man. Все игры в разработанной авторами системе происходят на дискретном поле размером 15×15 клеток; поле состоит из свободного пространства и стен и никогда не меняется. На игровом поле также расположен игрок и некоторые объекты разных цветов, поведение и характеристики которых определяются правилами, едиными для всех объектов одного цвета. Игра происходит в реальном времени, разделенном на отдельные промежутки; в каждый дискретный отрезок времени игрок и объекты могут двигаться на одну клетку в любом свободном направлении. Конец игры наступает по истечении определенного количества отрезков времени; игра считается «выигранной» по достижении некоторого числа очков.

Для кодирования подобных игр в системе используется набор переменных и две матрицы. Переменные задают базовые параметры игры — длительность игры, максимальное количество очков и число объектов, а также шаблон их движения. В первой матрице задаются эффекты от столкновений (то есть расположения на одной клетке) объектов разных цветов друг с другом и игроком. Список возможных эффектов включает в себя: исчезновение, перемещение на случайную клетку, а также пустой эффект. Например, в матрице может быть указано, что при столкновении красного и зеленого объекта красный объект исчезает, а зеленый объект перемещается на случайную клетку. Подобным образом во второй матрице описывается положительное или отрицательное изменение очков игрока от столкновения объектов и игрока.

Как и в случае описанной ранее системы Ludi, авторы ставили перед собой задачу генерации интересных игр с помощью методов генетического программирования. В системе использовался достаточно тривиальный генетический алгоритм, на каждой итерации которого копия текущей игры подвергалась операции мутации (то есть случайному изменению некоторых значений переменных и элементов матриц), после чего оригинальная игра заменялась на копию, если значение функции оценки новой игры было выше.

Особый интерес в данной работе представляет реализация функции оценки. По мнению психологов и игровых дизайнеров, одной из ключевых составляющих любой игры является изучение игроком ее правил; чем интереснее игроку осваивать игру и учиться лучше в нее играть, тем интереснее игра в целом. Основываясь на данной теории, авторы работы предложили оценивать интересность игр на основе их *изучаемости*. Таким образом, высоко оцениваются игры, которые, с одной стороны, достаточно сложны для новых игроков, но, с другой стороны, достаточно легки для освоения. Для автоматической оценки изучаемости игр авторы реализовали игрового агента на основе нейронной сети, обучение которого происходит с помощью алгоритма эволюционной

стратегии. Каждая новая игра несколько раз тестируется с помощью двух агентов со случайными стартовыми параметрами. Игра оценивается тем выше, чем больше очков было набрано агентами; если любым из агентов было набрано большое количество очков уже на первых тестах, игра считается слишком легкой и оценивается отрицательно.

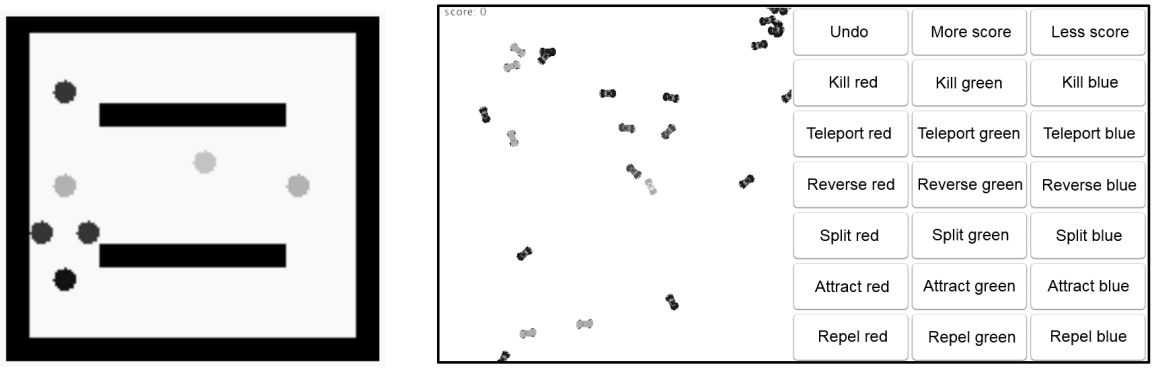


Рис. 7. Скриншоты простых аркадных игр, сгенерированных в рамках экспериментов Дж. Тогелиуса

Дальнейшие исследования авторы проводили в области интерактивной полуавтоматической генерации игр [21]. В реализованной ими системе моделируется простая двумерная игра, в которой игрок в реальном времени управляет автомобилем с простой физикой движения и может сталкиваться с управляемыми компьютером машинами разных цветов. На старте системы при столкновении игрока с другими машинами ничего не происходит; однако, после каждого столкновения игрок может нажать одну из кнопок, отвечающих за разные эффекты («добавить очки», «уничтожить объект», и так далее). После нажатия кнопки в игре происходит выбранный эффект, а система с помощью алгоритмов машинного обучения пытается определить, почему игрок нажал на эту кнопку, и сгенерировать соответствующие игровые правила. Скриншоты игр, сгенерированных в описанных выше экспериментах, изображены на рис. 7.

Во всех описанных в данной главе работах для генерации игровых правил в той или иной мере использовались генетические алгоритмы и методы эволюционного программирования. Недостатком данного подхода является необходимость генерировать и оценивать большое количество игр, из которых лишь единицы удовлетворяют заданным критериям качества.

Альтернативный подход был предложен в рамках проекта *Variations Forever* [19], в котором описание множества генерируемых игр реализовано на основе парадигмы *Answer Set Programming* (ASP, программирование стабильных моделей). В данном подходе множество всех возможных игр задается с помощью набора логических ограничений.

```
pushes(A,B) :- on_collide(A,B,bounce) , on_collide(B,A,bounce).
kills(A,B)  :- on_collide(A,B,kill).
indirectly_pushes(A,B) :- pushes(A,B).
indirectly_pushes(A,C) :- pushes(A,B) , indirectly_pushes(B,C).
winnable_via(indirect_push_kill(A,C)) :-
  indirectly_pushes(A,B),kills(B,C).
compute {
  player_agent(A) , goal(kill_all(B)),
  winnable_via(indirect_push_kill(A,B)) }.
```

Рис. 8. Набор логических ограничений для генерации игр в Variations Forever

Каждый такой набор задает некоторое множество игр, схожих по некоторому критерию. На рис. 8 приведен пример такого набора, описывающего все игры, выиграть в которых можно, толкая объекты друг в друга. Таким образом, вместо перебора и модификации конкретных игр, в данном подходе изменяется исходный набор ограничений. Добавление одного ограничения позволяет исключить из рассмотрения не отдельную игру, а целый класс неинтересных игр.

Конкретные игры генерируются из набора ограничений и представляются в виде *логических термов*; термами являются константы, переменные и функторы, аргументами которых являются другие логические термы. Игровое правило, представленное в виде логического терма, может иметь следующий вид:

$$\text{scripted\_event}(\text{spawn}(\text{boss\_creature},\ \text{temple}),\ 120) \qquad (1)$$

и означать событие «через две минуты после начала игры поместить в храм существо класса boss_creature». Подобным образом в Variations Forever представлены все параметры генерируемых игр: характеристики и поведение игровых объектов, игровое поле с препятствиями и эффекты взаимодействия с ними игрока. Пример полного ASP-описания простой игры и скриншот ее визуализации приведены на рис. 9.

space_resolution(32,24).
space_topology(spherical).
background(grids; stars).
active_agent(red; yellow; white; cyan).
agent_movement(red,asteroids; white,asteroids;
yellow,roguelike; cyan,pacman).
agent_population(red,many; white,singleton;
yellow,singleton; cyan,many).
agent_collide_effect(red,white,kill;
cyan,yellow,kill).
player_agent(white).
obstacle_distribution(enclosure; random_walls;
random_blocks).
obstacle_collide_effect(red,kill; white,kill).
goal(kill_all(red)).

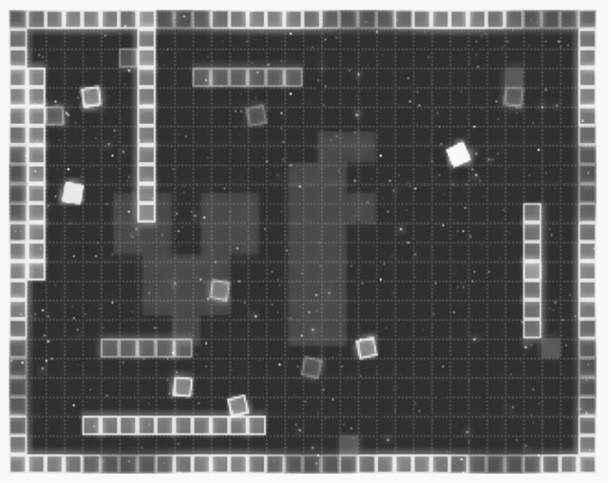


Рис. 9. Описание и скриншот игры, сгенерированной в системе Variations Forever

Отметим, что на данном этапе в системе отсутствует возможность автоматической оценки качества игр и корректировки набора ограничений; обе этих задачи должны решаться вручную игровым дизайнером.

Наиболее успешных результатов в области процедурной генерации видеоигр добился М. Кук в рамках проекта *ANGELINA*, каждая новая версия которого решает новую исследовательскую задачу. Скриншоты игр, сгенерированных разными версиями системы ANGELINA, представлены на рис. 10.

Первая версия системы ANGELINA генерировала простые двумерные игры с помощью эволюционного моделирования [7] и во многом была схожа с описанной выше работой Дж. Тогелиуса. Главным отличием данной системы является представление игровых правил, расположения объектов и карты уровня в виде трех раздельных сущностей, генерируемых с помощью методов *кооперативной коэволюции*. Под кооперативной коэволюцией в эволюционном моделировании понимают раздельное эволюционирование

особей из популяций различных классов, при котором, однако, оценка результатов происходит совместно. При совместной оценке принимается во внимание, насколько хорошо особи из разных популяций подходят друг другу. Например, в системе ANGELINA может быть сгенерирована интересная карта уровня и оригинальные игровые правила, которые будут оценены отрицательно вследствие того, что с такими правилами уровень становится непроходимым.

Рис. 10. Скриншоты игр, сгенерированных разными версиями системы ANGELINA (в хронологическом порядке)

Во второй версии системы ANGELINA авторы сфокусировались на генерации игр в так называемом жанре «Metroidvania» [4]. Данный жанр игр характеризуется сложной структурой уровней и наличием особых «предметов-усилителей», которые позволяют нашедшему их игроку открывать все большее число ранее недоступных ему областей карты. Примером усилителя может служить предмет, увеличивающий высоту прыжка игрока и тем самым позволяющий ему допрыгнуть до ранее недоступных участков уровня. Авторы добавили в систему кооперативной коэволюции отдельную популяцию с характеристиками усилителей, положительно оценивая такие наборы усилителей, которые позволяют проходить игру разными путями. Оценка игр в ANGELINA проводится автоматически, путем моделирования действий игрока.

Данный подход был развит в системе *Mechanic Miner*, в которой при генерации усилителей во внимание принимался исходный код игры [6]. Теперь, при генерации усилителя изменяемую им характеристику система Mechanic Miner выбирала не из заранее заданного списка, а из всех переменных, найденных с помощью рефлексии в исходном коде игры. Например, в процессе генерации одной из игр система Mechanic Miner обнаружила у объекта *player* переменную *acceleration,* и в качестве эффекта усилителя инвертировала значение ее y-компоненты:

$$\text{player.acceleration.y} \mathrel{*}= -1 \tag{2}$$

что в терминах игры означает, что после нахождения данного усилителя игрок буквально начнет ходить по потолку уровня, попросту игнорируя ранее мешавшие ему препятствия. Найденная с помощью Mechanic Miner в автоматическом режиме, данная игровая механика является крайне интересной, хотя и не новой — она лежит в основе популярной коммерческой игры *VVVVVV* (Cavanagh, 2010).

Отметим, что автоматическое изменение кода программы конечно же не может не приводить к ошибкам исполнения. Для решения этой проблемы, в процессе оценивания каждой игры Mechanic Miner перехватывает все возможные исключения и дает отрицательную оценку тем усилителям, что привели к появлению ошибок. В случае отсутствия

ошибок Mechanic Miner использует комплексную функцию оценки, принимающую в расчет *полезность* усилителя; усилитель считается полезным, если он позволяет игроку получить доступ к новым участкам карты и помогает пройти игру. Так, увеличение высоты прыжка на один пиксель скорее всего не даст игроку преимуществ, поэтому такой усилитель будет оценен отрицательно. У данного подхода есть несколько недостатков; во-первых, он малоприменим на уровнях большого размера из-за сложности расчета доступности различных участков карты; во-вторых, в текущей реализации отсутствует обработка «слишком полезных» усилителей, делающих игру неинтересной и тривиальной: например, усилитель, который моментально переносит игрока к концу уровня, получит высокую оценку несмотря на то, что он сводит к нулю все остальные составляющие игры́.

В последней версии ANGELINA авторы впервые реализовали процедурную генерацию трехмерных игр [3, 5]. Несмотря на смену архитектуры системы при ее портировании на платформу Unity, алгоритмы генерации игр практически не изменились.

Отдельный интерес представляют разработанные в рамках проекта ANGELINA методы генерации игр на заданную тему [2]. Принимая на входе одно слово или целый текст, ANGELINA выделяет ключевые слова и подбирает к ним различные ассоциации. Далее система анализирует семантику полученных слов с помощью нескольких источников и на основе полученных результатов производит поиск свободно распространяемых графических материалов и музыки, подходящих по смыслу и настроению к ключевым словам и ассоциациям.

Отметим, что система ANGELINA не лишена недостатков. Авторы указывают, что реализованная ими автоматическая оценка игр дает недостаточно качественные результаты, в то время как живые игроки затрудняются давать сравнительные оценки множеству похожих игр.

3. Языки описания игр

Одним из главных факторов успешности системы процедурной генерации является выбор способа представления контента, поскольку именно от него во многом зависит разнообразие генерируемого контента [25]. При выборе способа представления контента нужно найти баланс между слишком общим и слишком узким представлением. Чем шире класс контента, который можно описать с помощью выбранного способа представления, тем сложнее найти среди экземпляров этого класса подходящие качественные решения; в случае выбора слишком узконаправленного способа представления, экземпляры генерируемого контента будут отличаться лишь деталями.

Данная проблема особенно актуальна в случае с процедурной генерацией игр, ведь многие игры не похожи друг на друга – например, шахматы и Mario не имеют между собой ничего общего. Достаточно общим способом представления, подходящим для всех игр, является, например, язык программирования C++. Действительно, любую игру можно представить в виде программы на C++. Однако, данный способ представления задает чрезвычайно обширный класс контента: подавляющее большинство программ, которые можно написать на языке C++, не будут являться играми; более того, лишь малая часть из них не будет являться случайным набором инструкций.

Несмотря на то, что во всех работах, описанных во втором разделе данной статьи, авторы создавали свой способ представления игровых правил, многие ученые исследуют

возможность разработки специализированного языка для описания игровых правил. Создание подобного языка позволит не только отделить описание игры от используемых средств визуализации, оценки и генерации игр, но и упростит задачу разработки и взаимодействия универсальных игровых программ [8].

Одним из первых и наиболее известных языков описания игр является Game Description Language (GDL), разработанный в стэндфордском университете [13]. Данный язык был разработан в рамках исследований УИП и может использоваться для описания широкого класса пошаговых игр с полной информацией; большинство описанных на нем игр схожи с шахматами и другими настольными играми на доске. GDL основан на логике первого порядка и крайне подробен. Описание даже простейших игр на GDL занимает несколько страниц; так, для описания правил «крестиков-ноликов» потребуется около трех страниц. Это является одной из причин, по которой данный язык не используется в системах процедурной генерации, поскольку автоматическое генерация и оценка правил на столь сложном языке затруднительны. Существуют различные модификации данного языка, в том числе для поддержки игр с неполной информацией [20].

Одной из первых попыток создания языка для описания видеоигр является проект *Extensible Graphical Game Generator* [16], в рамках которого был реализован язык для описания двумерных графических игр. Основной целью данного проекта являлась разработка системы, позволяющей создавать игры, не прибегая к программированию. С помощью данной системы, пользователь может сгенерировать полноценную игру на языке Perl, просто описав ее правила. Отметим, что в данной работе не идет речь о процедурной генерации игр, поскольку все правила генерируемой игры должны быть заранее заданы пользователем.

Язык Video Game Description Language является попыткой реализовать универсальный язык для описания видеоигр, который бы мог использоваться как в УИП, так и в процедурной генерации игр [8]. Как и в работах, описанных во втором разделе этой статьи, на данном этапе создатели VGDL делают акцент на описании простых аркадных игр. Описание игр на VGDL состоит из пяти частей:

— описания карты на основе двумерного массива ASCII-символов;
— списка сопоставлений ASCII-символов с графическими спрайтами;
— набора спрайтов с описаниями их схем движения;
— набора правил взаимодействия игровых объектов друг с другом;
— набора условий конца игры.

На данный момент ведется активная разработка языка, его спецификации изменяются и улучшаются. Кроме того, для генерации описанных на данном языке игр была реализована система PyVGDL [18], генерирующая готовые игры по их описанию.

Примером более узконаправленного подхода к представлению правил является язык Strategy Game Description Language [14], предназначенный для описания стратегических видеоигр. Авторы языка предлагают использовать для описания различных игровых объектов и правил представление на основе деревьев, в котором сложность правил увеличивается с добавлением новых вершин. Данный язык находится на начальном уровне разработки и в данный момент поддерживает только описание простых характеристик игровых объектов.

Другим примером языка для описания одного жанра игр является язык Card Game Description Language [9], предназначенный для описания карточных игр. Данный язык

описывает три основных характеристики карточных игр: стадии игры, каждая из которых характеризуется своим набором правил, ранжирование карт и их комбинаций, а также условия победы. Описание правил игр основано на контекстно-свободной грамматике; допустимые правила включают в себя скрытие и раскрытие карт, передвижение карт между разными участками игрового стола, а также действия со ставками и так далее. Авторы CGDL планируют реализовать на его основе систему процедурной генерации карточных игр.

Заключение

Данная статья посвящена обзору исследований в области процедурной генерации игровых правил и игр различных жанров. На основе данного обзора можно сделать вывод, что задача процедурной генерации игр является на сегодняшний день актуальной. Существующие экспериментальные решения ориентированы на генерацию игр нескольких отдельных жанров, а созданные с их помощью игры зачастую крайне просты и не могут конкурировать с играми, созданными человеком.

Таким образом, перспективным направлением для дальнейших исследований является улучшение существующих и создание новых алгоритмов генерации с целью повышения качества и разнообразия генерируемых игр. В дальнейшем нами планируется работа над созданием системы процедурной генерации игр, не ограниченной рамками конкретных жанров.

Литература

1. Browne C. Evolutionary Game Design / C. Browne — Berlin: Springer, 2011. — 122 p.
2. Cook, M. Aesthetic Considerations for Automated Platformer Design / M. Cook, S. Colton, A. Pease // AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. — 2012. — P. 56–63.
3. Cook, M. Automating Game Design In Three Dimensions / M. Cook, S. Colton, J. Gow // Proceedings of the AISB Symposium on AI and Games. — 2014. — P. 20–24.
4. Cook, M. Initial Results from Co-operative Co-evolution for Automated Platformer Design / M. Cook, S. Colton, J. Gow // European Conference on Applications of Evolutionary Computing. — 2012. — P. 194–203.
5. Cook, M. Ludus Ex Machina: Building A 3D Game Designer That Competes Alongside Humans / M. Cook, S. Colton // Proceedings of Fifth International Conference on Computational Creativity. — 2014. — P. 54–62.
6. Cook, M. Mechanic Miner: Reflection-Driven Game Mechanic Discovery and Level Design / M. Cook, S. Colton, A. Raad, J. Gow // Applications of Evolutionary Computation. — 2013. — P. 284–293.
7. Cook, M. Multi-Faceted Evolution Of Simple Arcade Games / M. Cook, S. Colton // IEEE Conference on Computational Intelligence and Games. — 2011. — P. 289–296.
8. Ebner, M. Towards a Video Game Description Language / M. Ebner, J. Levine, S. Lucas, T. Schaul, T. Thompson, J. Togelius // Dagstuhl Seminar on Artificial and Computational Intelligence in Games. — 2013. — P. 1–17.
9. Font, J. A Card Game Description Language / J. Font, T. Mahlmann, D. Manrique, J. Togelius // Applications of Evolutionary Computation. — 2013. — P. 254–263.

10. Hendrikx, M. Procedural Content Generation for Games: A Survey / M. Hendrikx, S. Meijer, J. Van Der Velden, A. Iosup // ACM Transactions on Multimedia Computing, Communications, and Applications. — 2013. — Vol. 9, No 1. — P. 1–22.
11. Hom, V. Automatic Design of Balanced Board Games / V. Hom, J. Marks // Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference. — 2007. — P. 25–30.
12. Koller, D. Generating and Solving Imperfect Information Games / D. Koller, A. Pfeffer // Proceedings of the 14th international joint conference on Artificial intelligence. — 1995. — P. 1185–1192.
13. Love, N. General Game Playing: Game Description Language Specification / N. Love, T. Hinrichs, M. Genesereth — Stanford: Stanford University, 2006. — 25 p.
14. Mahlmann, T. Towards Procedural Strategy Game Generation: Evolving Complementary Unit Types / T. Mahlmann, J. Togelius, G. N. Yannakakis // Applications of Evolutionary Computation. — 2011. — P. 93–102.
15. Nelson, M. Recombinable Game Mechanics for Automated Design Support / M. Nelson, M. Mateas // Proceedings of the Fourth AIIDE Conference. — 2008. — P. 84–89.
16. Orwant, J. EGGG: Automated Programming for Game Generation / J. Orwant // IBM Systems Journal — 2000. — Vol. 39, No 3. — P. 782–794.
17. Pell, B. METAGAME in Symmetric Chess-Like Games / B. Pell // Heuristic Programming in Artificial Intelligence 3: The Third Computer Olympiad. — 1992. — P. 1277–1307.
18. Shaul, T. A Video Game Description Language for Model-based or Interactive Learning / T. Shaul // Proceedings of IEEE Conference on Computational Intelligence in Games. — 2013. — P. 1–8.
19. Smith, A. Variations Forever: Flexibly Generating Rulesets from a Sculptable Design Space of Mini-Games / A. Smith, M. Mateas // Proceedings of the IEEE Conference on Computational Intelligence and Games. — 2010. — P. 273–280.
20. Thielscher, M. A General Game Description Language for Incomplete Information Games / M. Thielscher // Proceedings of the AAAI Conference on Artificial Intelligence. — 2010. — P. 994–999.
21. Togelius, J. A Procedural Critique of Deontological Reasoning / J. Togelius // Proceedings of the 2011 DIGRA International Conference. — 2011. — P. 110–123.
22. Togelius, J. An Experiment in Automatic Game Design / J. Togelius, J. Schmidhuber // IEEE Symposium on Computational Intelligence and Games. — 2008. — P. 111–118.
23. Togelius, J. Characteristics of Generatable Games / J. Togelius, M. Nelson, A. Liapis // Proceedings of the Fifth Workshop on Procedural Content Generation in Games. — 2014. — P. 63–68.
24. Togelius, J. Procedural Content Generation in Games: A Textbook and an Overview of Current Research / J. Togelius, N. Shaker, M. Nelson — Berlin: Springer, 2014. — 142 p.
25. Togelius, J. Search-Based Procedural Content Generation: A Taxonomy and Survey / J. Togelius, G.N. Yannakakis, K.O. Stanley, C. Browne // IEEE Transactions on Computational Intelligence and AI in Games. — 2011. — Vol. 3, No 3. — P. 172–186.

Меженин Михаил Григорьевич, аспирант, преподаватель кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), m.mezhenin@gmail.com.

Поступила в редакцию 23 декабря 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 1, pp. 5–20

DOI: 10.14529/cmse150101

SURVEY ON PROCEDURAL GAME GENERATION

M.G. Mezhenin, South Ural State University (Chelyabinsk, Russian Federation)
m.mezhenin@gmail.com

Procedural content generation (PCG) is one of the most important fields of research in videogame industry. PCG allows creating different parts of games automatically; an interesting task of PCG is generating game rules and even whole games. In this paper, we present an overview of research in this field; algorithms from evolutionary computation and logic programming fields that can be used to generate game rules and games in different genres are described, as well as methods of evaluating the resulting games. We briefly describe general game-playing programs and the PCG systems that were used in them. Several representations and description languages utilized in PCG systems to encode game rules are also described.

Keywords: procedural content generation, game design, general game playing, artificial intelligence.

References

1. Browne C. Evolutionary Game Design. Berlin: Springer, 2011. 122 P.
2. Cook M. Aesthetic Considerations for Automated Platformer Design // AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. 2012. P. 56–63.
3. Cook M. Automating Game Design In Three Dimensions // Proceedings of the AISB Symposium on AI and Games. 2014. P. 20–24.
4. Cook M. Initial Results from Co-operative Co-evolution for Automated Platformer Design // European Conference on Applications of Evolutionary Computing. 2012. P. 194–203.
5. Cook M. Ludus Ex Machina: Building A 3D Game Designer That Competes Alongside Humans // Proceedings of Fifth International Conference on Computational Creativity. 2014. P. 54–62.
6. Cook M. Mechanic Miner: Reflection-Driven Game Mechanic Discovery and Level Design // Applications of Evolutionary Computation. 2013. P. 284–293.
7. Cook M. Multi-Faceted Evolution of Simple Arcade Games // IEEE Conference on Computational Intelligence and Games. 2011. P. 289–296.
8. Ebner M. Towards a Video Game Description Language // Dagstuhl Seminar on Artificial and Computational Intelligence in Games. 2013. P. 1–17.
9. Font J. A Card Game Description Language // Applications of Evolutionary Computation. 2013. P. 254–263.
10. Hendrikx M. Procedural Content Generation for Games: A Survey // ACM Transactions on Multimedia Computing, Communications, and Applications. 2013. Vol. 9, No 1. P. 1–22.
11. Hom V. Automatic Design of Balanced Board Games // Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference. 2007. P. 25–30.
12. Koller D. Generating and Solving Imperfect Information Games // Proceedings of the 14th international joint conference on Artificial intelligence. 1995. P. 1185–1192.

13. Love N. General Game Playing: Game Description Language Specification / N. Love T. Hinrichs M. Genesereth Stanford: Stanford University, 2006. 25 P.
14. Mahlmann T. Towards Procedural Strategy Game Generation: Evolving Complementary Unit Types // Applications of Evolutionary Computation. 2011. P. 93–102.
15. Nelson M. Recombinable Game Mechanics for Automated Design Support // Proceedings of the Fourth AIIDE Conference. 2008. P. 84–89.
16. Orwant J. EGGG: Automated Programming for Game Generation // IBM Systems Journal. 2000. Vol. 39, No 3. P. 782–794.
17. Pell B. METAGAME in Symmetric Chess-Like Games // Heuristic Programming in Artificial Intelligence 3: The Third Computer Olympiad. 1992. P. 1277–1307.
18. Shaul T. A Video Game Description Language for Model-based or Interactive Learning // Proceedings of IEEE Conference on Computational Intelligence in Games. 2013. P. 1–8.
19. Smith A. Variations Forever: Flexibly Generating Rulesets from a Sculptable Design Space of Mini-Games // Proceedings of the IEEE Conference on Computational Intelligence and Games. 2010. P. 273–280.
20. Thielscher M. A General Game Description Language for Incomplete Information Games // Proceedings of the AAAI Conference on Artificial Intelligence. 2010. P. 994–999.
21. Togelius J. A Procedural Critique of Deontological Reasoning // Proceedings of the 2011 DIGRA International Conference. 2011. P. 110–123.
22. Togelius J. An Experiment in Automatic Game Design // IEEE Symposium on Computational Intelligence and Games. 2008. P. 111–118.
23. Togelius J. Characteristics of Generatable Games // Proceedings of the Fifth Workshop on Procedural Content Generation in Games. 2014. P. 63–68.
24. Togelius J. Procedural Content Generation in Games: A Textbook and an Overview of Current Research / J. Togelius N. Shaker M. Nelson Berlin: Springer, 2014. 142 P.
25. Togelius J. Search-Based Procedural Content Generation: A Taxonomy and Survey // IEEE Transactions on Computational Intelligence and AI in Games. 2011. Vol. 3, No 3. P. 172–186.

Received December 23, 2014.

УДК 004.92, 004.75 DOI: 10.14529/cmse150102

УДАЛЕННАЯ ВИЗУАЛИЗАЦИЯ БОЛЬШИХ ОБЪЕМОВ ДАННЫХ

Д.В. Ненаженко, Г.И. Радченко

Вычислительные мощности и аппаратные характеристики персональных вычислительных устройств не всегда позволяют обеспечить должный уровень производительности для обеспечения визуализации больших объемов данных, возникающих в результате решения различных задач с использованием суперкомпьютерных вычислительных систем. Для обеспечения прозрачного и удобного доступа к таким данным может применяться подход удаленной визуализации, при котором клиент используется исключительно для отображения видео-информации с одного или нескольких удаленных серверов визуализации. В данной работе рассматриваются виды удаленной визуализации, используемые технологии, для обеспечения взаимодействия между клиентскими приложениями и удаленными серверами, анализируются различные подходы к решению задачи удаленной визуализации.

Ключевые слова: удаленная визуализация, сервисы визуализации, удаленный рендеринг.

Введение

Применение суперкомпьютерного моделирования для решения задач из различных секторов промышленности и экономики позволяет значительно повысить качество производимой продукции, уменьшить затраты на проектирование и разработку новых товаров и обеспечить новое качество услуг, предоставляемых пользователям. Такие задачи возникают во множестве сфер, включая машиностроение [7], моделирование биологических систем [8, 13], компьютерные игры [2] и др. Современные системы, обеспечивающие решение задач такого типа, обычно, представляют собой комплексы ПО, состоящие из нескольких взаимосвязанных программных систем, обменивающихся данными в процессе выполнения задачи. Результатом, обычно, является визуальное представление данных, интересующих пользователя. Однако данные, получаемые в результате решения таких задач, могут представлять собой файлы, размер которых может достигать от десятков гигабайт до нескольких терабайт.

Вычислительные мощности и аппаратные характеристики персональных вычислительных устройств не всегда позволяют обеспечить должный уровень производительности для обеспечения визуализации результатов таких задач. Также, большие объемы данных тяжело передавать по сети, от сервера клиенту и обратно. При этом, если для визуализации требуется не весь объем данных, а лишь некоторая их часть, значительно уменьшается эффективность использования доступных сетевых ресурсов. Таким образом, на сегодняшний день, проблема обработки и визуализации больших объемов данных является одной из важнейших проблем оперативного получения и использования информации. Под большими данными подразумеваются такие данные, обработка которых на стандартных устройствах конечного пользователя занимает недопустимое для работы время. Распространение мобильных устройств, рост объема результатов инженерных вычислений заострили проблему отсутствия единого универсального подхода к решению проблемы удаленной визуализации.

Удаленная визуализация (*remote visualization, remote rendering*) — это подход к визуализации, при котором клиент (устройство с относительно низкими вычислительными

возможностями) используется для отображения массивов данных, визуализированных на одном или нескольких удаленных серверах визуализации [15]. Удаленная визуализация данных может применяться в различных задачах математической обработки, как правило, это обработка трехмерной графики [6], статичных моделей, различных анимированных сцен, смоделированных заранее, визуализация в реальном времени (например, трехмерное сканирование) [4], обработка различного рода статистических данных и др.

Анализ существующих задач, решаемых на суперкомпьютерных вычислительных системах и методов удаленного доступа к суперкомпьютерным ресурсам показал, что удаленная визуализация является *актуальным* направлением исследований, так как высокое развитие суперкомпьютерных вычислений и совершенствование сетевых технологий позволяет пользователям отдавать предпочтения легким, маломощным портативным устройствам для доступа к удаленным вычислительным ресурсам.

Целью данной работы является анализ существующих протоколов, платформ и систем удаленной визуализации. Дальнейший текст статьи организован следующим образом. В первом разделе описан процесс удаленной визуализации и ее виды. Во втором разделе представлены протоколы управления, используемые при удаленной визуализации. В третьем разделе рассмотрены наиболее распространенные на сегодняшний день платформы удаленной визуализации. В заключении приводится краткий анализ результатов работы.

1. Удаленная визуализация

Удаленная визуализация данных (см. рис. 1) позволяет реализовать процесс визуализации на базе удаленных центров обработки данных, передавая конечному пользователю непосредственный результат визуализации в виде изображений либо видеоряда. При использовании удаленной визуализации, процесс обработки данных разделяется на различные по сложности этапы, которые могут быть реализованы как на сервере, так и на клиенте. В зависимости от того, как распределены эти этапы между клиентом и сервером, система удаленной визуализации может обеспечить балансировку нагрузки на вычислительные мощности клиентских и серверных систем, а также минимизацию сетевого трафика между ними.

Рис. 1. Удаленная визуализация данных

Для использования систем удаленной визуализации, обычно, от клиентской машины не требуется наличия больших вычислительных мощностей. Однако необходимо высоко-

скоростное сетевое соединение для доступа клиента к ресурсам сервера удаленной визуализации. Сервер, в свою очередь, обеспечивает решение поставленных вычислительных задач, пре-рендеринг и рендеринг готового изображения.

Рендеринг (rendering, визуализация) — в компьютерной графике, процесс преобразования цифровых моделей в визуализируемое представление — изображение [3].

Пре-рендеринг — это предварительные операции по подготовке виртуальной сцены, обеспечивающие оптимизацию дальнейшего процесса рендеринга [3]. Различные системы удаленной визуализации могут выполнять различные действия с виртуальной сценой в ходе этапа пре-рендеринга. Например, при пре-рендеринге может происходить подготовка карты теней, предварительная отрисовка предыдущих либо следующих кадров, подготовка текстур модели.

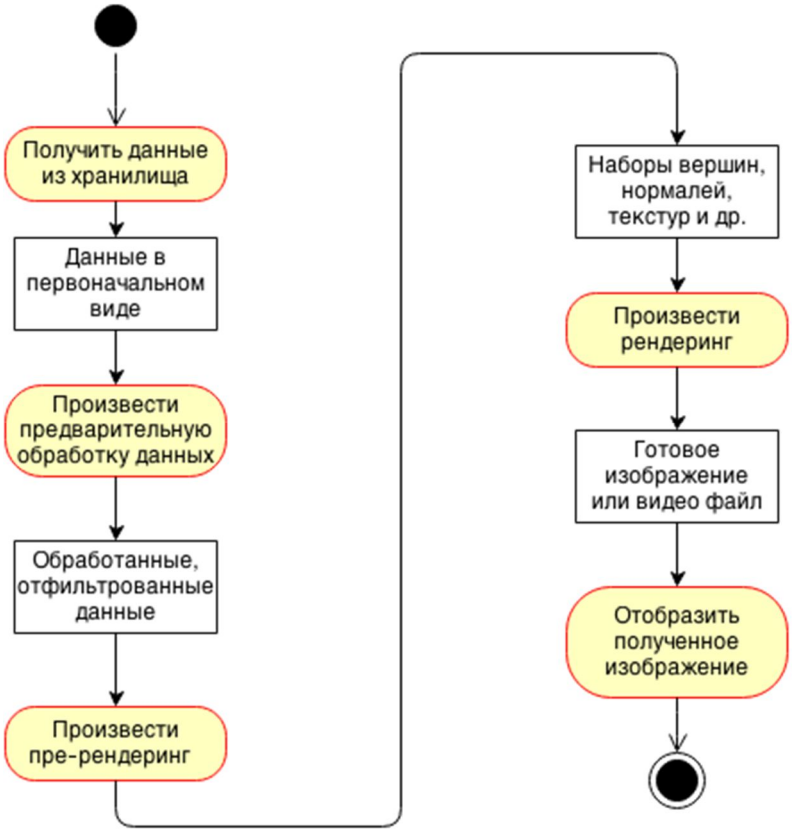


Рис. 2. Процесс удаленной визуализации

На рис. 2 изображен процесс обработки данных в системах удаленной визуализации. Действия в потоке могут быть реализованы различными способами и выполняться как на серверной части, так и на клиентской, в зависимости от реализации.

Начальные данные из хранилища извлекаются в необработанном виде. Такие «сырые» данные могут содержать в себе много лишней информации, не требующей визуализации, поэтому производится предварительная обработка этих данных, включающая очистку от лишней информации, агрегацию и преобразование данных к необходимому для визуализации виду. Обработанные данные необходимо преобразовать в визуализируемый набор, представляющий собой вершины, нормали, текстуры и др., после чего произвести пре-рендеринг. После того как данные возможно визуализировать происхо-

дит рендеринг данных в готовое изображение или видео файл. Конечный результат отображается у пользователя.

В табл. 1 представлены варианты размещения различных этапов удаленной визуализации на серверной или клиентской стороне.

Подход *«чистой удаленной визуализации»* обеспечивает выполнение всех расчетов и обработку изображения на серверной части системы удаленной визуализации, с последующей передачей изображения клиенту. На клиенте никакой обработки результата не производится, обеспечивается только отображение результата с сервера.

Таблица 1

Виды удаленной визуализации

Операция	**Чистая удаленная визуализация**	**Смешанная удаленная визуализация**	**Обработка данных на сервере**	**Хранение данных на сервере**
Отображение	клиент	клиент	клиент	клиент
Рендеринг	сервер	клиент	клиент	клиент
Пре-рендеринг	сервер	сервер	клиент	клиент
Обработка данных	сервер	сервер	сервер	клиент
Хранилище данных	сервер	сервер	сервер	сервер

К достоинствам данного подхода можно отнести:

- низкие требования к производительности клиентского устройства;
- возможность гибкого применения алгоритмов сжатия полученных изображений/видео потока, включая алгоритмы с потерей данных;
- возможность использования браузера в качестве клиентского ПО;
- редко требует установки дополнительного программного обеспечения.

С другой стороны, основным недостатком такого подхода являются высокие требования к качеству Интернет-соединения для передачи видеопотока. Интерактивное взаимодействие с удаленной системой, реализуемое посредством постоянной передачи видеопотока (особенно в высоком разрешении) требует малых значений времени ожидания пакетов, высокой скорости обмена и высокой надежности связи между клиентской и серверной машиной.

Смешанная удаленная визуализация реализуется в том случае если от сервера клиенту передается набор данных подготовленных к рендерингу (примитивы, нормали, текстуры и др.), а на клиентской машине происходит финальный рендеринг данных и их отображение. Так же, на клиентской машине может производиться наложение удаленного изображения на объекты, доступные только на локальной машине (например, в рамках приложений дополненной реальности).

При таком подходе к удаленной визуализации, требования к качеству сетевого соединения не такие жесткие как при чистой удаленной визуализации. Также, появляется возможность совместной визуализации удаленных и локальных данных.

К недостаткам такого подхода можно отнести необходимость иметь на клиентской машине отдельный графический процессор, отвечающий за рендеринг данных. Также, для работы такой системы скорее всего потребуется установка и настройка дополнительного ПО.

Еще одним подходом является *серверная обработка данных.* При таком подходе на сервере происходит только хранение «сырых» данных и их обработка (очистка от лишней информации, агрегация и преобразование данных к необходимому для визуализации виду).

Основным достоинством такого подхода к удаленной визуализации является то, что серверной части не требуется наличие графического процессора, за счет этого можно увеличить производительность CPU или использовать GPU как сопроцессор.

Из недостатков стоит отметить дополнительные ограничения на клиентскую часть системы, так как необходимость иметь мощный графический процессор, а также специализированное ПО для визуализации данных, может означать дополнительные расходы. Еще одним недостатком данного подхода является то, что объем передаваемых по сети данных может сильно колебаться, для этого необходимо высокоскоростное сетевое оборудование.

В случае, когда сервер выступает только в роли хранилища данных, а вся обработка ведется на клиенте, используется подход, предполагающий *серверное хранение данных.* Такой подход часто не относят к удаленной визуализации, так как вся обработка и визуализация происходит на клиентской машине.

Основным достоинством является простота реализации серверной части. Также такой подход не требует больших мощностей от сервера, достаточно иметь высокоскоростной доступ к носителям информации. Такой подход удобен в случаях, когда время обращения к локальному хранилищу клиентской машины значительно превосходит время обращения к удаленному хранилищу.

Однако надо отметить, что подход обладает следующими недостатками:

- необходимость предустановленного на клиентской машине дополнительного программного обеспечения;
- необходимость высокопроизводительного оборудования на клиентской стороне;
- необходимость наличия высокоскоростного интернет соединения, для передачи больших объемов «сырых» данных.

Основной задачей построения системы удаленной визуализации является нахождение баланса между вычислительными мощностями клиентского оборудования и пропускной способностью сети. На сегодняшний день развитие высокоскоростных соединений, а так же развитие суперкомпьютерных вычислений позволяет максимально эффективно использовать подходы чистой и смешанной удаленной визуализации. Подходы, ориентированные исключительно на хранение и обработку данных на сервере часто не относят к методам удаленной визуализации, подразумевая что в этом случае, фактическая задача визуализации решается на клиентском устройстве.

2. Протоколы удаленного управления

Для интерактивного взаимодействия клиента и сервера удаленной визуализации необходимы специальные *протоколы управления*. Они обеспечивают передачу команд удаленного управления от клиента удаленному графическому интерфейсу, а также передачу готового изображения на клиентскую машину.

Одним из первых и широко распространенных протоколов удаленного управления является протокол RFB [12] (Remote Framebuffer Protocol), базирующийся на протоколе TCP. RFB появился в 1998 г. при создании тонкого клиента в рамках проекта Videotile, после чего нашел применение с появлением технологии VNC в 2002 г. [12]. RFB работает на уровне кадрового буфера, а значит его можно применять для графических оконных систем, например X Window System, Windows, Quartz Compositor.

В начале своего развития, RFB был относительно простым протоколом, основанным на графических примитивах, таких как «положить прямоугольник пиксельных данных на заданную координатами позицию». При этом, сервер посылает небольшие прямоугольники изображения клиенту при изменении их содержимого. Такая подход потребляет значительный трафик, так как прямоугольные участки изображения передаются клиенту без сжатия. В более новых версиях протокола RFB было реализовано несколько методов кодирования передаваемых прямоугольников, обеспечивающих снижение нагрузки на канал. Самый простой метод кодирования, поддерживаемый всеми клиентами и серверами — «raw encoding» (сырое кодирование), при котором пиксели передаются в порядке слева-направо, сверху-вниз, и после передачи первоначального состояния экрана передаются только изменившиеся пиксели. Этот метод работает очень хорошо при незначительных изменениях изображения на экране (движения указателя мыши по рабочему столу, набор текста под курсором), но загрузка канала становится очень высокой при одновременном изменении большого количества пикселей, например, при просмотре видео в полноэкранном режиме. За время своего развития протокол оброс различными дополнительными функциями и опциями, такими как передача файлов, сжатие, безопасность. В качестве примера можно рассмотреть Remote Rendering Protocol [1], включающий в себя размеры экрана, параметры освещения, глубина цвета и др. Также протокол RRP включает в себя возможность работы с multi-touch устройствами, что необходимо для работы с мобильными устройствами, такими как планшетные компьютеры и смартфоны.

Еще одним известным представителем протоколов удаленного управления является RemoteFX [11] (RDP 7.1). Данный протокол является разработкой компании Microsoft и основан на ранее разработанном ими же протоколом RDP. RDP более ранних версий подразумевал частичный рендеринг изображения на машине клиента, то есть часть информации передавалась удаленно, а часть отрисовывалась на графических процессорах клиента. Чаще всего, данный протокол использовался для обеспечения доступа к удаленным серверам. С появлением Windows 8, заявленной для работы с портативными устройствами, такими как планшеты, Microsoft пересмотрели на этот счет некоторые моменты, и в новой версии протокола на клиентское устройство передается только изображение. Данный протокол позволяет эффективно взаимодействовать с протоколами передачи видео и аудио.

ICA [10] — это закрытый протокол для сервера приложений, разработанного компанией Citrix Systems. Протокол был разработан с учетом соединений с низкой пропускной

способностью, что сделало его более надежным. Протокол ICA позволяет передавать представление текстовых экранов, представление графических экранов Windows-приложений, аудио/видео потоки, ввод с клавиатуры и мыши не зависимо от платформы (поддерживаются платформы Windows, Macintosh, Linux, и другие OS семейства UNIX). Развитием концепции ICA стал набор технологий под названием HDX (High Definition eXperience) [14], представленный компанией Citrix в 2013 г. и направленный на предоставление конечным пользователям возможности работы с удаленными вычислительными системами в режиме «высокой точности», не зависимо от того, с какого устройства или через какой тип соединения обеспечивается доступ. Данная технология обеспечивает бесшовный режим (когда с сервера передаются данные не о всём рабочем столе, а только о содержимом конкретного окна), который позволяет работать как на тонких клиентах (у пользователя появляется одно приложение без прочего окружения — меню пуск, рабочего стола), так и на «толстых» клиентах, позволяя интегрировать удалённые приложения в локальный рабочий стол (все приложения, кроме одного, работают локально, одно работает на сервере, но пользователь не замечает разницы между ними). HDX состоит из следующих ключевых компонентов:

- *HDX MediaStream* — обеспечивает доставку мультимедийного контента, включая видео, звук;
- HDX Broadcast — позволяет передавать мультимедийный контент практически в любых условиях, эффективно сжимает и кэширует данные;
- HDX Plug-n-Play — обеспечивает взаимодействие виртуальной ОС с различными plug-n-play устройствами;
- HDX RichGraphics — обеспечивает эффективную передачу трехмерной графики на клиентскую машину.

Проведенный анализ протоколов удаленного управления для реализации систем удаленной визуализации показывает, что в настоящее время существует целый ряд решений, направленных на обеспечение эффективного визуального взаимодействия с удаленными вычислительными системами. При этом, за последнее десятилетие появились протоколы, предоставляющие возможность работы не только с простейшими диалоговыми оконными приложениями, но и удаленного взаимодействия со сложными графическими интерфейсами.

3. Платформы и сервисы удаленной визуализации

Платформа удаленной визуализации — это промежуточное программное обеспечение, предоставляющее механизмы для визуализации данных на удаленных вычислительных системах. Платформы удаленной визуализации обеспечивают взаимосвязь между клиентскими приложениями и серверными, предоставляя доступ к графическим ускорителям и координируя взаимодействие команд от клиента серверу.

Платформы удаленной визуализации предоставляют средства для реализации систем удаленной визуализации включая пользовательский интерфейс для удаленной визуализации, сервисы управления сеансами визуализации (на базе виртуальных машин), интерфейсы для взаимодействия между визуализаторами и графическими ускорителями, протоколы удаленного управления.

На сегодняшний день существует множество реализаций различных подходов к удаленной визуализации. Одной из наиболее распространенных и широко известных платформ является *ParaViewWeb Framework* [5]. Проект данной платформы удаленной визуализации был начат в 2000 г. совместными усилиями компаний Kitware Inc. и Los Alamos National Laboratory. Позднее, компания Kitware на базе разрабатываемой платформы начала реализацию веб-ориентированной системы удаленной визуализации. Данная система была финансирована исследовательским отделом армии США, и в 2002 г. проект перешел в промышленную эксплуатацию. Данный проект является открытым, и любой разработчик моежет привнести свой вклад в развитие данного проекта. Система не зависит от аппаратной платформы и может взаимодействовать с различными облачными сервисами.

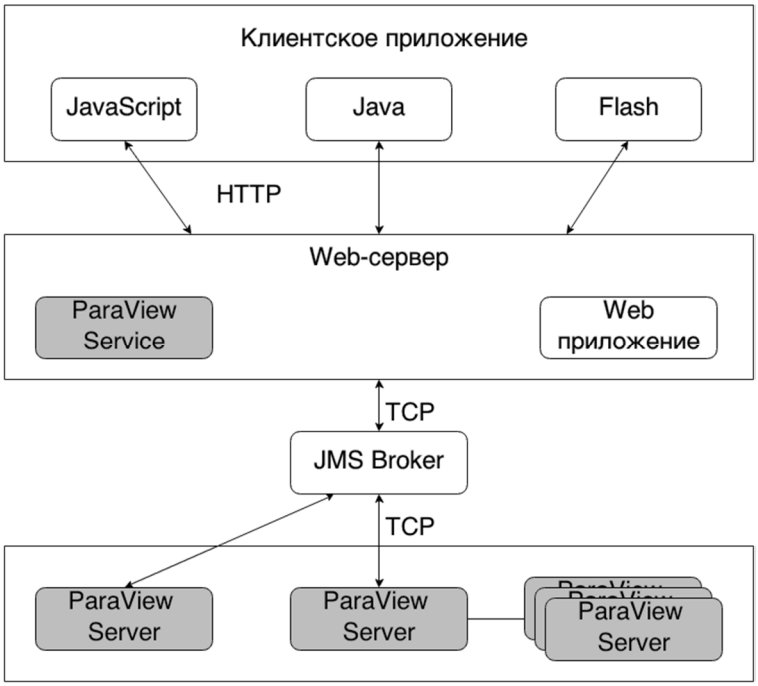


Рис. 3. Архитектура приложения на базе ParaViewWeb Framework

На рис. 3 представлен пример архитектуры приложения, обеспечивающего удаленную визуализацию на платформе ParaViewWeb Framework. Серым выделены компоненты, предоставленные платформой, белым — сторонние системы и системы, разработанные для конкретного пользователя. Для обеспечения взаимодействия клиента и сервера применяются стандартные компоненты, такие как JMS (Java Message Service), а также протоколы HTTP и TCP. В случае необходимости, можно использовать различные надстройки, например для обеспечения шифрования данных, сжатия данных без потерь и др. Для доступа к удаленной системе визуализации, на клиентской машине достаточно иметь web-браузер с поддержкой JavaScript или Flash.

На базе Paraview было создано множество систем удаленной визуализации и Paraview считается своеобразным эталоном среди подобных систем.

Платформа удаленной визуализации *NICE DCV* [9] — это инструмент для создания систем удаленной трехмерной и двумерной визуализации, основанных на предоставлении удаленного доступа к специально сгенерированным виртуальным машинам, обеспе-

чивающим удаленный запуск и работу с приложениями визуализации. Данная платформа базируется на механизме виртуальных машин, что обеспечивает хорошую масштабируемость. Системы, реализованные на базе NICE DCV не зависят от платформы, и могут работать со смешанными сессиями (Windows, Linux) на одном узле в рамках разных виртуальных машин.

Для работы с NICE DCV, пользователь должен инициализировать виртуальную машину визуализации, указав, какие наборы данных он желает визуализировать, а также какие пакеты визуализации ему необходимы. NICE DCV создает экземпляр соответствующей виртуальной машины и представляет пользователю удаленный графический доступ к ней. Таким образом, пользователю предоставляется полноценная виртуальная машина, в рамках которой он может получить полноценный доступ к интересующим его данным, располагающимся на удаленной вычислительной системе и проанализировать их, используя знакомые «настольные» приложения для визуализации и анализа данных.

Платформа NICE DCV предоставляет набор компонент, таких как DCV OpenGL Library, для обеспечения взаимодействия приложения удаленной визуализации и графического процессора, DCV Protocol, для связи между клиентским устройством и графическим ядром серверного приложения, и др. Также стоит отметить, что платформа NICE DCV предоставляет возможность реализовывать системы удаленной визуализации с использованием различных подходов (например, чистая или смешанная удаленная визуализация).

Заключение

В представленной работе нами были проанализированы различные протоколы и платформы удаленной визуализации данных. Анализ протоколов удаленной визуализации показал, что одной из наиболее эффективных на сегодняшний день технологий удаленной визуализации является технологий HDX, разработанный компанией Citrix. Он обеспечивает возможность эффективного использования ресурсов удаленных вычислительных машин в режиме «удаленного рабочего стола», независимо от платформы клиента и метода его соединения с сетью. Также, нами были рассмотрены наиболее распространенные на сегодняшний день платформы удаленной визуализации. В то время как ParaViewWeb Framework обеспечивает возможность внедрения методов удаленной визуализации в собственные приложения, коммерческая платформа NICE DCV предоставляет готовое решение, поддерживающее большую степень настройки под требования конечного пользователя. Таким образом, разработка системы предоставления готовых виртуальных машин, предварительно настроенных на решение задачи удаленной визуализации результатов пользовательских расчетов на основе существующих прикладных пакетов визуализации данных можно считать перспективным направлением исследований.

Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных научных исследований (грант № 14-07-00420) и Совета по грантам Президента Российской Федерации (номер проекта МК-7524.2015.9).

Литература

1. Aumüller, M. Remote Hybrid Rendering of Exascale Data for Immersive Virtual Environments // EASC — April 2013. URL: http://www.easc2013.org.uk/sites/default/files/Pdfs/ParallelSession3a/Martin_Aumueller_-_Remote_Hybrid_Rendering_-_EASC_2013.pdf (дата обращения 02.06.2014).
2. Boukerche, A. Remote rendering and streaming of progressive panoramas for mobile devices / A. Boukerche, R. Pazzi // Proceedings of the 14th annual ACM international conference on Multimedia. — USA: ACM. 2006. — P. 691–694.
3. Brooker, D. Essential CG Lighting Techniques with 3ds Max / D. Brooker // Focal Press: Elsevier Inc. — 2008. — P.398.
4. Chen, B.T. A 3D scanning system based on low-occlusion approach / B.T. Chen, W.S. Lou, C.C. Chen, H.C. Lin // Proceedings of Second International Conference on 3-D Digital Imaging and Modeling. — USA: IEEE, 1999. — P. 506–515.
5. Documentation ParaViewWeb. Kitware Inc. URL: http://www.paraview.org/ParaView3/Doc/Nightly/www/js-doc/index.html (дата обращения 20.05.2014).
6. Evans, A. 3D graphics on the web: A survey / A. Evans, M. Romeo, A. Bahrehmand // Computers & Graphics. — 2014. Vol. 41. — P. 43–61.
7. Hwang, K. Multiprocessor Supercomputers for Scientific/Engineering Applications / K. Hwang// Computer. — 1985. — Vol. 18, No. 6. — P. 57–73.
8. Kikinis, R. Computer-assisted interactive three-dimensional planning for neurosurgical procedures / R. Kikinis, P.L. Gleason, T.M. Moriarty // Neurosurgery. — 1996. — Vol. 38, No. 4. — P 640-651.
9. Official web-resource NICE software. Раздел посвященный DCV. URL: http://www.nice-software.com/products/dcv (дата обращения 13.06.2014)
10. Protocol specification ICA. ICA Functional specifications. URL: http://publications.europa.eu/tenders/our/documents/ao_10017/cd_cordis/annexes/applications_doc/phasing_out/ica/lot2_fsd_ica_functional_specifications_v030.pdf (дата обращения 20.05.2014).
11. Protocol specification Microsoft RemoteFX. URL: http://msdn.microsoft.com/en-us/library/ff635423.aspx (дата обращения 15.03.2014)
12. Richardson, T. The RFB Protocol / T. Richardson, K. Wood // ORL, Cambridge. — January 1998. URL: http://www.realvnc.com/docs/rfbproto.pdf (дата обращения 22.03.2014).
13. Tomandl, B. F. Local and Remote Visualization Techniques for Interactive Direct Volume Rendering in Neuroradiology / B. F. Tomandl, P. Hastreiter, C. Rezk-Salama // Radiographics. — 2001. — Vol. 21, No. 6. — P. 1561–1572.
14. White paper of Citrix HDX. URL: https://www.citrix.ru/content/dam/citrix/en_us/documents/products-solutions/citrix-hdx-technologies.pdf (дата обращения 15.03.2014).
15. Zellmann, S. Image-Based Remote Real-Time Volume Rendering: Decoupling Rendering From View Point Updates / S. Zellmann, M. Aumüller, U. Lang // ASME 2012 International Design Engineering Technical Conferences and Computers and Information in En-

gineering Conference. — USA: American Society of Mechanical Engineers, 2012. — P. 1385–1394.

Ненаженко Дмитрий Владимирович, аспирант кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), nenazhenkodv@susu.ac.ru.

Радченко Глеб Игоревич, к.ф.-м.-н., доцент кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), gleb.radchenko@susu.ru.

Поступила в редакцию 2 октября 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 1, pp. 21–32

DOI: 10.14529/cmse150102

REMOTE VISUALIZATION OF LARGE DATA SETS

D.V. Nenazhenko, South Ural State University (Chelyabinsk, Russian Federation)
nenazhenkodv@susu.ac.ru,
G.I. Radchenko, South Ural State University (Chelyabinsk, Russian Federation)
gleb.radchenko@susu.ru

Computing power and hardware specifications of personal computing devices often cannot provide adequate performance to ensure the visualization of large amounts of data. Such data sets can be provided as results of various supercomputing experiments. To ensure a transparent and user-friendly access to such data one can use remote visualization approach. Remote visualization concept determines that the client is used only to display information from one or more remote visualization servers. This paper discusses types of the remote visualization technology used to provide interaction between client applications and remote servers, different ways of solving the problem of the remote visualization.

Keywords: remote visualization, visualization services, remote rendering.

References

1. Aumüller, M. Remote Hybrid Rendering of Exascale Data for Immersive Virtual Environments // EASC — April 2013. URL: http://www.easc2013.org.uk/sites/default/files/Pdfs/ParallelSession3a/Martin_Aumueller_-_Remote_Hybrid_Rendering_-_EASC_2013.pdf (accessed 02.06.2014).
2. Boukerche A., Pazzi R. Remote rendering and streaming of progressive panoramas for mobile devices // Proceedings of the 14th annual ACM international conference on Multimedia. USA: ACM. 2006. P. 691–694.
3. Brooker, D. Essential CG Lighting Techniques with 3ds Max / D. Brooker // Focal Press: Elsevier Inc. — 2008. — P.398.

4. Chen B.T., Lou W.S., Chen C.C., Lin H.C. A 3D scanning system based on low-occlusion approach // Second International Conference on 3-D Digital Imaging and Modeling. USA: IEEE, 1999. P. 506–515.
5. Documentation ParaViewWeb. Kitware Inc. URL: http://www.paraview.org/ParaView3/Doc/Nightly/www/js-doc/index.html (accessed 20.05.2014)
6. Evans A., Romeo M., Bahrehmand A. 3D graphics on the web: A survey // Computers & Graphics. 2014. Vol. 41. P. 43–61.
7. Hwang K. Multiprocessor Supercomputers for Scientific/Engineering Applications. // Computer. 1985. Vol. 18, No. 6. — P. 57–73.
8. Kikinis R., Gleason P.L., Moriarty T.M. Computer-assisted interactive three-dimensional planning for neurosurgical procedures // Neurosurgery. 1996.
9. Official web-resource NICE software, DCV. URL: http://www.nice-software.com/products/dcv (accessed 13.06.2014).
10. Protocol specification ICA. ICA Functional specifications. URL: http://publications.europa.eu/tenders/our/documents/ao_10017/cd_cordis/annexes/applications_doc/phasing_out/ica/lot2_fsd_ica_functional_specifications_v030.pdf (accessed 20.05.2014).
11. Protocol specification Microsoft RemoteFX. URL: http://msdn.microsoft.com/en-us/library/ff635423.aspx (accessed 15.03.2014).
12. Richardson T. Wood K. The RFB Protocol // ORL, Cambridge. — January 1998. URL: http://www.realvnc.com/docs/rfbproto.pdf (accessed 22.03.2014).
13. Tomandl B.F., Hastreiter P., Rezk-Salama C. Local and Remote Visualization Techniques for Interactive Direct Volume Rendering in Neuroradiology // Radiographics. 2001. Vol. 21, No. 6. P. 1561–1572.
14. White paper of Citrix HDX. URL: https://www.citrix.ru/content/dam/citrix/en_us/documents/products-solutions/citrix-hdx-technologies.pdf (accessed 15.03.2014).
15. Zellmann S., Aumuller M., Lang U. Image-Based Remote Real-Time Volume Rendering: Decoupling Rendering From View Point Updates // ASME 2012 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. USA: American Society of Mechanical Engineers. 2012. P. 1385–1394.

Received October 2, 2014.

УДК 004.457, 004.382.2 DOI: 10.14529/cmse150103

ОБ ОДНОМ ПОДХОДЕ К МОДЕЛИРОВАНИЮ СУПЕРКОМПЬЮТЕРНЫХ КОМПЛЕКСОВ[1]

П.А. Швец, Вад.В. Воеводин, С.И. Соболев

В НИВЦ МГУ предложен подход к созданию системы контроля автономного функционирования суперкомпьютерных комплексов на основе графовой модели суперкомпьютера. С использованием данного подхода была реализована система контроля Octotron, которая сейчас проходит апробацию в суперкомпьютерном центре МГУ. Данная статья описывает проблемы и задачи, с которыми столкнулись авторы при реализации данной системы и ее запуске на суперкомпьютерах «Чебышёв» и «Ломоносов». Рассматриваются выбранные и разработанные авторами программные инструменты для работы с графами, кратко описывается язык, используемый для описания модели, затрагиваются вопросы визуализация модели и импорта данных мониторинга.

Ключевые слова: суперкомпъютер, модель суперкомпьютера, мониторинг, инструменты программирования, автономное функционирование, надежность.

Введение

Поддержка автономного функционирования суперкомпьютерного центра — одна из важнейших задач, с которой сталкиваются их держатели и администраторы. Этот же вопрос встал перед нами в рамках работ по обеспечению эффективной надежной работы Суперкомпьютерного комплекса МГУ. Однако анализ мировой практики поддержки суперкомпьютерных центров показал, что каждый решает эту задачу по-своему, создавая индивидуальные и непереносимые комплексы.

Нами был предложен [1] метод контроля работы суперкомпьютерного комплекса на основе модели его функционирования, представленной в виде расширенного мультиграфа. Вершины графа описывают физические (ЦПУ, кондиционеры др.) и логические (область подкачки, файловая система и др.) компоненты комплекса. Ребра графа описывают связи между компонентами: например, вершина «стойка» может быть связана с вершиной «шасси» связью «содержит», вершина «кондиционер» — связана с вершиной «горячий коридор» связью «охлаждает». Пример простейшей модели показан на рис. 1. С вершинами и связями модели ассоциируются атрибуты, описывающие состояние компонент (температура, IP-адрес и др.), правила, позволяющие преобразовывать атрибуты (например, вычислять скорость изменения характеристик) и реакции, срабатывающие, когда атрибуты принимают определенное значение (например, информирование системного администратора). Разработанная нами система контроля, основываясь на описанной модели, производит получение, обработку и анализ реальных данных с различных аппаратных и программных сенсоров, а с помощью набора правил и реакций происходит контролирование штатной работы комплекса. Данный подход позволяет обрабатывать ряд ситуаций, которые сложно отслеживать в рамках раздельного мониторинга и контроля всех компонент, при котором не учитываются физические и логические связи между разными компонентами. Например, при проблемах с одним источником питания можно от-

[1] Статья рекомендована к публикации программным комитетом Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: многообразие суперкомпьютерных миров».

ключить только то оборудование, которое питается непосредственно от него, или же следить за количеством узлов в конкретной очереди с ошибками определенного типа. Также наличие модели со связями полезно для систематизации знаний о суперкомпьютерных комплексах и потенциальных источниках проблем. Каждый объект рассматривается как потенциально сбойный. Становится возможным оценить, как сбой каждого конкретного компонента повлияет на соседние компоненты и на работу комплекса в целом. Подробней принципы моделирования описаны в статье [1]. Настоящая работа посвящена детальному описанию реализованного нами подхода к моделированию суперкомпьютерных комплексов.

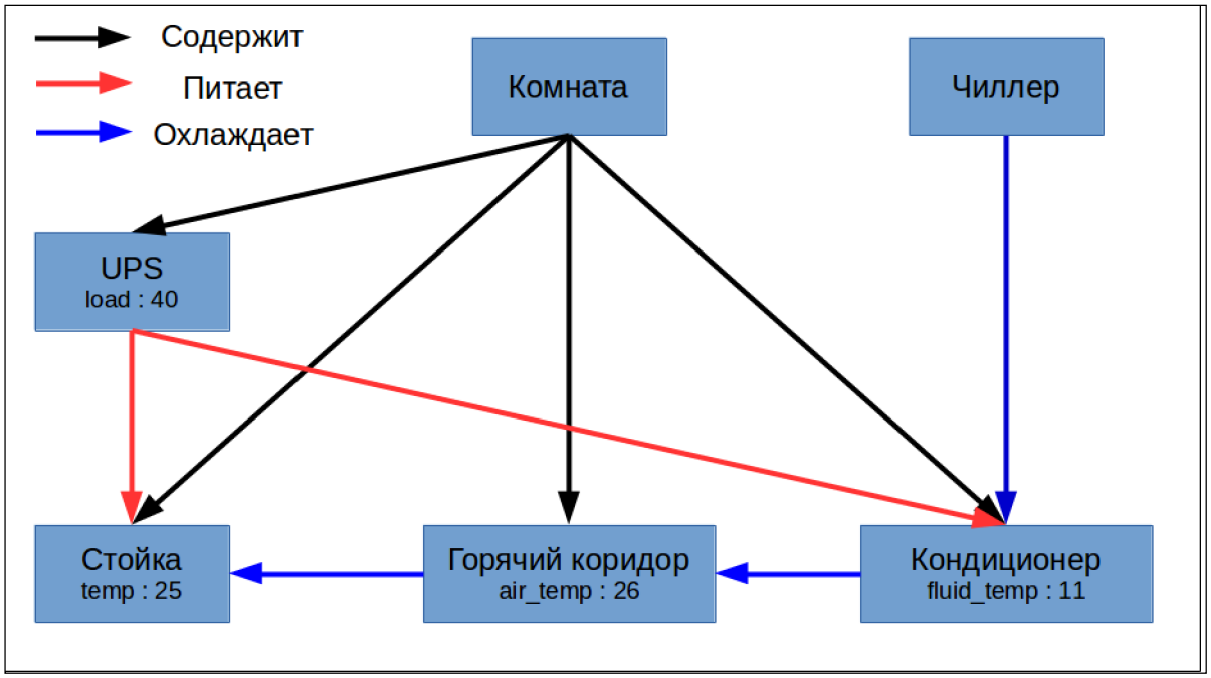


Рис. 1. Пример части модели суперкомпьютера

Статья организована следующим образом. В разделах 1 и 2 рассматриваются высокоуровневые и низкоуровневые инструменты для работы с графами. В разделах 3 и 4 описываются и обосновывается выбранный нами подход к разработке комплекса. В разделе 5 описывается используемый язык описания модели и приводятся примеры описания. В 6 разделе рассматривается методика построения полной модели суперкомпьютера. Раздел 7 посвящен вопросу визуализации построенной модели суперкомпьютера. В разделе 8 описан подход к управлению системой. Раздел 9 посвящен настройке средств мониторинга на суперкомпьютерах «Чебышёв» и «Ломоносов» для использования в разработанной системе контроля. В заключении суммируются основные результаты работы и описываются планы дальнейшего развития системы.

1. Анализ средств для работы с графами

На основе изучения характеристик современных суперкомпьютерных комплексов и формирования базовых принципов построения их моделей были сформулированы требования к программной системе работы с графами, которая должна лежать в основе разрабатываемого нами программного средства:

- Быстрая работа с большими графами. Сотни тысяч вершин, миллионы атрибутов — таков, по нашей оценке, масштаб требуемых графов для описания моделей суперкомпьютеров верхних строчек списка Top500.
- Поддержка кратных ребер. Некоторые объекты модели могут быть связаны двумя и более типами связей. Например, вершина «стойка» должна быть связана с вершиной «шасси» связями «содержит» и «питает».
- Поддержка атрибутов разных типов (строки, целые числа, числа с плавающей запятой) для вершин и ребер.

Нами были изучены известные средства для работы с большими графами, такие как Gephi [2] и Tulip/porgy [3].Gephi больше ориентирован на интерактивную работу пользователя с графами, а Tulip/porgy, имеющий средства для автоматического преобразования графов, использовал неподходящую нам систему правил преобразований. Доработка этих инструментов была признана нецелесообразной. Было принято решение об исследовании более низкоуровневых средств.

2. Анализ низкоуровневых библиотек и программных средств для работы с графами

Существует огромное множество библиотек для работы с графами на самых разных языках программирования, поэтому мы рассмотрели наиболее популярные из них и выбрали удовлетворяющую нашим требованиям.

- Boost Graph Library [4] (BGL) — популярная библиотека, отвергнутая нами из-за ее основного языка программирования — C++. Требования к разрабатываемой системе корректировались в ходе разработки, что могло бы потребовать очень больших временных затрат при использовании C++ как основного языка проекта.
- Graph-tool [5] — интерфейс для BGL на языке программирования Python. Согласно предварительной оценке, производительности языка не хватило бы для обработки требующихся нам объемов данных.
- Neo4j [6] — графовая СУБД, написанная на языке Java и ориентированная на работу с высоконагруженными сервисами.

3. Технологии реализации

Для разработки нами был выбран язык Java, к достоинствам которого можно отнести возможность быстрой разработки и наличие множества готовых библиотек, и база данных Neo4j как средство для работы с графами. Neo4j — нереляционная база данных, оперирующая понятиями «узлы» (nodes), «связи» (relationships) и «атрибуты» (properties), которые могут устанавливаться как на узлы, так и на связи. Это полностью подходит под наши требования к описанию модели. Кроме того, Neo4j обеспечивает следующие полезные возможности:

- Механизм автоматического кэширования позволяет хранить часто используемые данные в памяти, а редко используемые — на диске. Данная особенность позволяет эффективно работать с графами очень больших размеров, что может понадобиться для описания суперкомпьютеров экзафлопсного масштаба.
- Быстрый поиск по свойствам (индексирование) напрямую влияет на скорость исполнения запросов к системе, а значит, и на максимальное количество обрабатываемых запросов в секунду, позволяя обрабатывать данные от большего числа сенсоров с большей частотой.

- ACID-транзакции [7] — данная модель транзакций облегчает техническую обработку параллельных запросов.
- Готовый механизм сохранения и загрузки с диска позволяет продолжать работу системы после остановки без потери прошлых данных.

4. Интерфейс работы с графами (API)

Мы предусмотрели возможность для работы не только с Neo4j, реализовав свой промежуточный интерфейс для работы с графами и используя Neo4j как одну из возможных реализаций. Если в какой-то момент нам понадобится функционал, отсутствующий в Neo4j, но присутствующий в другой системе, мы сможем перейти на нее без существенного изменения логики работы системы. Исходя из наших требований, API для работы с графом реализует следующие возможности:

- Создать объект.
- Создать связи между созданными объектами.
- Установить и получить атрибуты по объекту или связи.
- Удалить заданные объекты, связи или атрибуты.

API для поиска по графу реализует следующие возможности:

- Найти все объекты/связи, у которых есть атрибут с заданным именем.
- Найти все объекты/связи, у которых есть атрибут с заданным именем и значением.
- Найти все объекты/связи, у которых есть строковые атрибуты с заданным именем и значением, подходящим под заданный шаблон.

На основе этих двух API мы реализовали все внутренние сервисы, позволяющие сохранять в модели более сложные объекты, обеспечивающие возможность обновления атрибутов, правил, вызов реакций, и реализующие внешний протокол запросов к графу.

5. Язык описания модели

После разработки системы встал вопрос об описании самой модели. Наиболее простой метод — описание модели на языке Java с помощью разработанного API. К сожалению, этот подход имеет существенный недостаток: Java — довольно «многословный» язык, а для описания модели хватило бы лишь небольшого подмножества языка. Также сама необходимость знать и использовать для описания язык Java может восприниматься неоднозначно.

Была предпринята попытка разработки собственного языка описания модели, который бы потом транслировался в Java. Первая версия преобразовывала код построчно с помощью набора регулярных выражений, но вскоре стало понятно, что ни развивать, ни поддерживать такой вариант невозможно: требовались все более сложные языковые конструкции, а их реализация с помощью регулярных выражения получалась очень сложной, громоздкой и неэффективной.

На следующем этапе мы изучили готовые средства, генерирующие трансляторы по формальной грамматике языка (например, ANTLR [8]), но после исследования наработок по этому направлению решили, что полноценная реализация собственного описательного языка займет слишком много времени.

В итоге в качестве основного языка описания моделей был выбран язык Python [9] и использован специальный интерпретатор Jython [10], исполняющий код на JVM (виртуальная машина, на которой исполняются Java-программы) и позволяющий без каких-либо дополнительных усилий использовать Java-классы из программы, написанной на Python.

Мы разработали модуль, предоставляющий простой интерфейс для исходного API на языке Java, и предоставляем пользователям примеры и документацию для создания своих моделей.

Рассмотрим, как с помощью предложенного подхода создаются модели. Первая часть описания модели — задание атрибутов, правил и реакций.

- Описание атрибутов совпадает с заданием словаря на языке Python в формате «имя : значение» для константных атрибутов или «имя : тип (значение по умолчанию)» для изменяющихся атрибутов (сенсоров). Значение по умолчанию может отсутствовать и тогда атрибут будет использоваться только после первого обновления. Атрибуты могут быть целыми и вещественными числами, строками и булевыми константами True/False.
- Правила описываются в стиле словаря, в формате «имя : конструктор правила». Имя обозначает имя атрибута, который будет создаваться правилом. Конструкторы правил индивидуальны в каждом случае и полностью описаны в документации [14].
- Реакции описываются в стиле словаря, в формате «условие реакции : конструктор реакции». Реакция срабатывает, когда выполняется требуемое условие, наиболее часто используемое условие - когда соответствующий атрибут принимает указанное значение. В качестве реакции мы создали 4 стандартных типа (Info, Warning, Danger, Critical), но у пользователя есть возможность модифицировать реакции, добавляя вызов произвольных скриптов или создавая собственные реакции.

```
eth_module : {
  "const" : {
    "interface" : "eth0"
  }
  "sensor" : {
    "recieve_errors" : Long(),
    "duplex" : String()
  }
  "var" : {
    "error_speed" : Speed("recieve_errors"),
    "low_errors" : UpperThreshold("error_speed", 10)
  }
  "react" : {
    Eqauls("low_errors", False).Delay(60) :
      Warning("tag", "NETWORK")
        .Msg("descr", "{interface}: errors growing fast in last minute"),
    NotEquals("duplex", "full") :
      Danger("tag", "NETWORK") :
        .Msg("descr", "{interface}: duplex mode changed: {duplex}")
  }
}
```

Рис. 2. Пример описания модуля для проверки сетевого интерфейса

Набор атрибутов, правил и реакций может быть объединен в модуль. Модули переносимы и могут быть использованы для создания нескольких разных моделей, имеющих схожие части. Мы предоставляем пользователям набор готовых модулей, соответствующих некоторым популярным средствам и протоколам мониторинга (SNMP, collectd). Пример описания модуля приведен на рис. 2. Данный модуль проверяет, что режим интерфейса (duplex) соответствует требуемому и что ошибки приема не растут быстрее, чем 10

штук в секунду на протяжении минуты (небольшой рост или кратковременные всплески могут присутствовать, но они не является опасным).

Вторая часть описания модели — создание объектов и связей.

- Для создания объектов используется функция CreateObjects(count, modules...) и CreateObject(modules...). Параметрами данных функций являются модули из описанных выше атрибутов, правил и реакций, а также количество объектов, которые необходимо создать. Есть возможность описывать необходимые модули в самом вызове функции. Первая функция возвращает список объектов, а вторая — один объект.
- Для создания связей между объектами используется набор функций OneToOne, OneToEvery, AllToAll и другие (описание каждой функции доступно в документации [14]). Параметрами этих функций являются сущности, которые надо соединить, и список типов связей.

```
# создание объектов
room      = CreateObject()
chiller   = CreateObject()
ups       = CreateObject({"sensor" : {"load" : Long()}})
fan       = CreateObject({"sensor" : {"fluid_temp" : Long()}})
hot_aisle = CreateObject({"sensor" : {"air_temp" : Long()}})
rack      = CreateObject({"sensor" : {"temp" : Long()}})

# создание связей между компонентами
OneToOne(room, ups      , "contains")
OneToOne(room, fan      , "contains")
OneToOne(room, hot_aisle, "contains")
OneToOne(room, rack     , "contains")
OneToOne(room, chiller  , "contains")
OneToOne(room, chiller  , "contains")

OneToOne(ups, fan , "power")
OneToOne(ups, rack, "power")

OneToOne(chiller  , fan      , "chill")
OneToOne(fan      , hot_aisle, "chill")
OneToOne(hot_aisle, rack     , "chill")
```

Рис. 3. Описание модели, изображенной на рис. 1, с помощью разработанного API

На рис. 3 приведен пример описания модели, соответствующей фрагменту, изображенному на рис. 1.

6. Методика построения модели

Суперкомпьютеры состоят из множества различных компонент, что существенно затрудняет построение полной описывающей модели. Однако чем больше компонент будет внесено в такую модель, тем более полный контроль над комплексом будет у системы.

На данный момент мы описываем следующие подсистемы суперкомпьютера (в скобках указаны примеры объектов из этой подсистемы):

- Система электропитания (источники бесперебойного питания, модули с батареями).
- Система охлаждения (чиллеры, кондиционеры, мониторинг среды).
- Управляющая часть (узлы доступа и компиляции, очереди задач).
- Вычислительная часть (шасси, узлы, диски, память).

- Файловая система (зависит от типа ФС).
- Ethernet сеть (коммутаторы, порты).
- Infiniband сеть (коммутаторы, менеджер сети).

Модель можно описывать вручную или же частично генерировать программно, пользуясь довольно регулярной структурой некоторых компонент. Индивидуальные данные объектов (ip-адрес, серийный номер и др.) в таком случае подгружаются из внешних CSV-файлов.

Одним из направлений работы над проектом является разработка инструментария для автоматизированного построения модели [15]. Сгенерированная модель будет требовать дальнейшей ручной доработки, но часть рутинной работы будет уже выполнена.

7. Визуализация модели

В ходе разработки регулярно возникала необходимость визуально оценить созданную модель. Изображение модели может быть полезно как для эмпирической проверки корректности построения, так и для демонстрационных или обучающих целей. Однако полностью отобразить модель практически нереально — алгоритмы расположения графов на плоскости не справляются с графами такого масштаба или показывают совершенно нечитаемый результат.

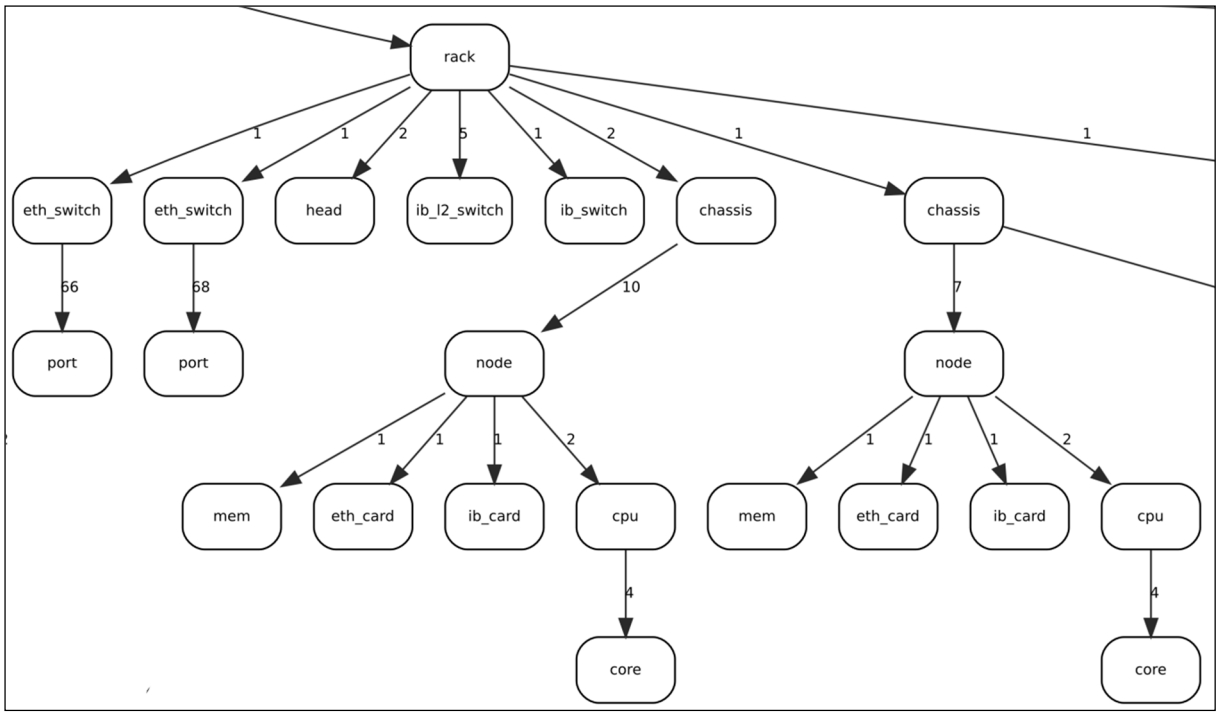


Рис. 4. Пример визуализации агрегированной модели

Но если не ставить задачу визуализации полного графа, а использовать регулярность многих компонент графа, то задача визуализации имеет решение. По многим типам связей модель выглядит как дерево с наличием похожих поддеревьев, которые можно разбивать по уровням и группировать, отображая в итоге метаграф из сгруппированных подграфов.

Был разработан прототип веб-сервиса, который производит «агрегацию» объектов модели и отображает получившийся метаграф. Пример результата его работы можно видеть на рис. 4 — это визуализация части модели суперкомпьютера «Чебышёв» [11].

Цифры на связях показывают, сколько «похожих» поддеревьев было объединено в одну вершину.

8. Работа с системой

После описания модели можно произвести запуск самой системы контроля и настроить импорт реальных данных. Управление системой происходит с помощью http-запросов к разработанной компоненте. Данный метод используется и для автоматического импорта данных в модель и для ручного управления. Такой подход позволяет управлять всем мониторингом из обычного веб-браузера и может легко интегрироваться в другие веб-сервисы, например, для визуализации текущего состояния суперкомпьютерного комплекса.

9. Наполнение системы реальными данными

Для нужд оперативного контроля информацию от суперкомпьютера можно распределить по трем категориям:

1. Информация от инфраструктурного оборудования — охлаждение, питание и прочее. Такой информации достаточно мало, однако именно она требует наиболее оперативной обработки и реагирования для сохранения оборудования комплекса в целостности. Требуемое время оповещения и реагирования — минуты. Также здесь используется механизм «активных событий» — при некоторых событиях данные в модель заносятся сразу при наступлении события, в обход стандартного механизма опроса.
2. Служебная информация от основных систем суперкомпьютера — файловая система, система очередей, служебные узлы. Данной информации значительно больше, ошибки могут повлечь проблемы с доступом к суперкомпьютеру, но не могут навредить оборудованию. Требуемое время оповещения и реагирования — от нескольких минут, до десятков минут.
3. Служебная информация с узлов — локальные проблемы на узлах, выход узлов из строя. Ошибки, которые могут проявляться на этом уровне, повлияют лишь на некоторые пользовательские задачи и не могут нанести серьезный вред оборудованию. Требуемое время оповещения и реагирования — десятки минут, часы.

Поток информации первой и второй категории не превысит нескольких сотен обновлений атрибутов в секунду, а поток информации третей категории не имеет фиксированных требований к частоте съема, которую можно регулировать для стабильной работы системы.

Таблица

Частота съема данных мониторинга

	«Чебышёв» (≈600 узлов)	«Ломоносов» (≈5000 узлов)
1-я категория	Опрос 1 раз в минуту Активные события	Опрос 1 раз в минуту Активные события
2-я категория	Опрос 1 раз в минуту	Опрос 1 раз в минуту
3-я категория	Опрос 1 раз в минуту	Опрос 1 раз в 10 минут

На данный момент система, работающая на среднем компьютере и графе, помещающемся в память, обрабатывает около 15 тысяч обновлений атрибутов в секунду. Исходя из этих данных, на системах «Чебышёв» и «Ломоносов» суперкомпьютерного комплекса МГУ мы установили частоту съема данных, указанную в таблице.

Заключение

В статье рассмотрен и обоснован выбор программных инструментов для работы с графами. Описан предложенный подход к описанию моделей суперкомпьютеров, предложен новый метод визуализации граф, получаемых из моделей. В статье описаны детали настройки разработанной системы на реальных суперкомпьютерах.

Система внедрена в опытную эксплуатацию в Суперкомпьютерном комплексе МГУ. Разработанная система доступна под открытой MIT лицензией [13, 14].

В дальнейшем мы планируем развивать средства визуализации и автоматической генерации модели и расширить библиотеку стандартных компонент суперкомпьютеров и средств мониторинга для облегчения процесса создания модели.

Работа выполнена при финансовой поддержке РФФИ, грант №12-07-33047.

Литература

1. Антонов, А.С. Разработка принципов построения и реализация прототипа системы обеспечения оперативного контроля и эффективной автономной работы суперкомпьютерных комплексов / А.С. Антонов, В.В. Воеводин, Вад.В. Воеводин и др. // Вестник УГАТУ. — 2014. — Т. 18, № 2. — С. 227–236.
2. Bastian, M. Gephi: an open source software for exploring and manipulating networks. / M. Bastian, S. Heymann, M. Jacomy // International AAAI Conference on Weblogs and Social Media. — 2009. — Vol. 8. — P. 361–362.
3. Pinaud, B. PORGY: A Visual Graph Rewriting Environment for Complex Systems. / B. Pinaud, G. Melançon, J. Dubois // Computer Graphics Forum — Eurographics Conference on Visualization (EuroVis 2012) special issue. — 2012. — Vol. 31. — P. 1265–1274.
4. BGL Library Documentation. URL: http://boost.org/libs/graph/doc/ (дата обращения: 29.12.2014).
5. GraphTool Program Description. URL: http://graph-tool.skewed.de/ (дата обращения: 29.12.2014).
6. Neo4j DBMS Description. URL: http://neo4j.org (дата обращения: 29.12.2014).
7. Gray, J. The Transaction Concept: Virtues and Limitations. / J. Gray. // Proceedings of the 7th International Conference on Very Large Databases. — 1981. — P. 144–154.
8. ANTLR Parser Generator Description ANTLR. URL: http://antlr.org (дата обращения: 29.12.2014).
9. Python Programming Language Description. URL: http://python.org (дата обращения: 29.12.2014).
10. Jython Interpreter Description. URL: http://jython.org (дата обращения: 29.12.2014).
11. Антонов, А.С. СКИФ МГУ — основа Суперкомпьютерного комплекса Московского университета / А.С. Антонов // Вторая Международная научная конференция «Суперкомпьютерные системы и их применение»: доклады конференции (27–29 октября 2008, Минск). — ОИПИ НАН Беларуси, 2008. — С. 7–10.

12. Воеводин, В.В. Практика суперкомпьютера «Ломоносов» / В.В. Воеводин, С.А. Жуматий, С.И. Соболев и др. // Открытые системы. — 2012. — № 7. — С. 36–39.
13. Ядро системы Octotron. URL: https://github.com/srcc-msu/octotron_core (дата обращения: 29.12.2014).
14. Рабочее окружения для создания модели системы Octotron. URL: https://github.com/srcc-msu/octotron (дата обращения: 29.12.2014).
15. Воеводин, Вад.В. Автоматическое определение и описание сетевой инфраструктуры суперкомпьютеров / В.В. Воеводин, К.С. Стефанов // Вычислительные методы и программирование: Новые вычислительные технологии. — 2014. — Т. 15, № 3. — С. 560–568.

Швец Павел Артёмович, программист, Научно-исследовательский вычислительный центр, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), shvets.pavel.srcc@gmail.com

Воеводин Вадим Владимирович, к.ф.-м.н., научный сотрудник, Научно-исследовательский вычислительный центр, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), vadim@parallel.ru

Соболев Сергей Игоревич, к.ф.-м.н, старший научный сотрудник, Научно-исследовательский вычислительный центр, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), sergeys@parallel.ru

Поступила в редакцию 26 декабря 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 1, pp. 33–43

DOI: 10.14529/cmse150103

AN APPROACH TO MODELING OF SUPERCOMPUTING CENTERS

P.A. Shvets, Research Computing Center, Moscow State University (Moscow, Russian Federation) shvets.pavel.srcc@gmail.com,
Vad.V. Voevodin, Research Computing Center, Moscow State University (Moscow, Russian Federation) vadim@parallel.ru,
S.I. Sobolev, Research Computing Center, Moscow State University (Moscow, Russian Federation) sergeys@parallel.ru

An approach to implementation of supercomputing center control system based on supercomputer graph model has been proposed in RCC MSU. The Octotron system has been developed on the basis of this approach, which is being tested in MSU Supercomputing Center currently. The article describes challenges and tasks, encountered by authors while developing and running the system on supercomputers «Chebyshev» and «Lomonosov». It also includes overview of graph tools used, brief description of modeling language, model visualization and monitoring data import.

Keywords: supercomputer, supercomputer model, monitoring, programming tools, autonomous functioning, resiliency.

References

1. Antonov A.S., Voevodin V.V., Voevodin Vad.V. Razrabotka principov postroenija i realizacija prototipa sistemy obespechenija operativnogo kontrolja i jeffektivnoj avtonomnoj raboty su-perkomp'juternyh kompleksov [Principles of Development System for Operational Control and Efficient Autonomous Work of Supercomputers]. Vestnik UGATU [Bulletin of UGATU]. 2014. Vol. 18. No. 2. P. 227–236.
2. Bastian M., Heymann S., Jacomy M. Gephi: an open source software for exploring and manipulating networks. International AAAI Conference on Weblogs and Social Media. 2009. Vol. 8. P. 361–362.
3. Pinaud B., Melançon G., Dubois J. PORGY: A Visual Graph Rewriting Environment for Complex Systems. Computer Graphics Forum — Eurographics Conference on Visualization (EuroVis 2012) special issue. 2012. Vol. 31. P. 1265–1274.
4. BGL Library Documentation. URL: http://boost.org/libs/graph/doc/ (accessed: 29.12.2014).
5. GraphTool Program Description. URL: http://graph-tool.skewed.de/ (accessed: 29.12.2014).
6. Neo4j DBMS Description. URL: http://neo4j.org (accessed: 29.12.2014).
7. Gray J. The Transaction Concept: Virtues and Limitations. Proceedings of the 7th International Conference on Very Large Databases. 1981. P. 144–154.
8. ANTLR Parser Generator Description. URL: http://antlr.org (accessed: 29.12.2014).
9. Python Programming Language Description. URL: http://python.org (accessed: 29.12.2014).
10. Jython Interpreter Description. URL: http://jython.org (accessed: 29.12.2014).
11. Antonov A.S. SKIF MGU - osnova Superkomp'juternogo kompleksa Moskovskogo universiteta [SKIF MSU — the Foundation of Moscow University Supercomputing Center]. Vtoraja Mezhdunarodnaja nauchnaja konferencija «Superkomp'juternye sistemy i ih primenenie»: doklady konferencii (27-29 oktjabrja 2008, Minsk) [Second Internation Conference «Supercomputing Systems and Applications»: papers (27–29 October 2008, Minsk)]. OIPI NAN Belarusi, 2008. P. 7–10.
12. Voevodin V.V., Zhumatij S.A., Sobolev S.I. Praktika superkomp'jutera «Lomonosov» [Practice of «Lomonosov» Supercomputer]. Otkrytye sistemy [Open Systems]. 2012. No. 7. P. 36–39.
13. Jadro sistemy Octotron [Octotron System Core]. URL: https://github.com/srcc-msu/octotron_core (accessed: 29.12.2014).
14. Rabochee okruzhenija dlja sozdanija modeli sistemy Octotron [Octotron Model Creation Environment]. URL: https://github.com/srcc-msu/octotron (accessed: 29.12.2014).
15. Vad.V. Voevodin, K.S. Stefanov. Avtomaticheskoe opredelenie i opisanie setevoj infrastruktury superkomp'juterov [Automatic Detection and Description of a Supercomputer Network Infrastructure]. Vychislitel'nye metody i programmirovanie: Novye vychislitel'nye tehnologii [Computational methods and programming: new computational technologies]. 2014. Vol. 15, No. 3. P. 560–568.

Received December 26, 2014.

УДК 004.657 DOI: 10.14529/cmse150104

ДЕКОМПОЗИЦИЯ ОПЕРАЦИЙ ПЕРЕСЕЧЕНИЯ И СОЕДИНЕНИЯ НА ОСНОВЕ ДОМЕННО-ИНТЕРВАЛЬНОЙ ФРАГМЕНТАЦИИ КОЛОНОЧНЫХ ИНДЕКСОВ

Е.В. Иванова, Л.Б. Соколинский

Статья посвящена вопросам декомпозиции реляционных операций путем использования распределенных колоночных индексов с доменно-интервальной фрагментацией. Такая декомпозиция позволяет организовать параллельное выполнение ресурсоемких реляционных операций без обменов данными между процессорными ядрами. Все фрагменты колоночного индекса хранятся в оперативной памяти в сжатом виде. При параллельном выполнении реляционной операции упакованные фрагменты индексов входных отношений загружаются на различные процессорные ядра, где происходят их распаковка, выполнение реляционной операции над фрагментами и упаковка частичного результата, представляющего собой наборы ключей. Затем частичные результаты объединяются в результирующий набор ключей, с использованием которого СУБД собирает результирующее отношение. Указанный подход позволяет организовать эффективное параллельное выполнение запросов к сверхбольшим базам данных на современных кластерных вычислительных системах, оснащенных многоядерными ускорителями.

Ключевые слова: сверхбольшие базы данных, параллельная обработка запросов, колоночные индексы, доменно-интервальная фрагментация, декомпозиция реляционных операций.

Введение

В настоящее время научно-практическая деятельность человека выдвигает все новые масштабные задачи, требующие обработки сверхбольших баз данных. Согласно прогнозам аналитической компании IDC, количество данных в мире удваивается каждые два года и к 2020 г. достигнет 44 Зеттабайт, или 44 триллионов гигабайт [1]. При этом современные технологии баз данных не могут обеспечить обработку столь крупных объемов данных. По оценке IDC в 2013 г. из всего объема существующих данных потенциально полезны 22%, из которых менее 5% были подвергнуты анализу. К 2020 году процент потенциально полезных данных может вырасти до 35%, преимущественно за счет данных от встроенных систем.

Фактически единственным эффективным решением проблемы хранения и обработки сверхбольших баз данных является использование параллельных систем баз данных, обеспечивающих распределенную обработку запросов на многопроцессорных вычислительных системах с распределенной памятью [2–6].

Традиционным подходом к организации хранения баз данных является строковое представление данных. Однако при выполнении типичных аналитических запросов к таблицам требуется считывать только небольшую часть полей в строках этих таблиц, поэтому строковое представление в этом случае оказывается неэффективным. Причиной этого является считывание с диска «лишних» полей в дополнение к тем полям, которые необходимы в данном запросе [7]. Возможным решением этой проблемы может быть использование механизмов колоночного представления данных, позволяющих получить на порядок лучшую производительность при обработке аналитических запросов [8]. Колоночное представление данных заключается в том, что данные хранятся не по строкам, а

по колонкам. Это означает, что с точки зрения SQL-клиента данные представлены в виде таблиц, но физически эти таблицы являются совокупностью колонок, каждая из которых представляет собой таблицу из одного поля. Дополнительным преимуществом колоночного представления является возможность использования эффективных алгоритмов сжатия данных, поскольку в одной колонке таблицы содержатся данные одного типа. Сжатие может привести к повышению производительности на порядок, поскольку меньше времени занимают операции ввода-вывода.

В последние годы основным способом наращивания производительности процессоров является увеличение количества ядер, а не тактовой частоты, и эта тенденция, вероятно, сохранится [9]. Сегодня GPU (Graphic Processing Units) и Intel MIC (Many Integrated Cores) значительно опережают традиционные процессоры в производительности по арифметическим операциям и пропускной способности памяти, позволяя использовать сотни процессорных ядер для выполнения десятков тысяч потоков. Последние исследования показывают, что многоядерные ускорители могут эффективно использоваться для обработки запросов к базам данных в оперативной памяти [10–12].

В соответствие с этим актуальной является задача разработки новых эффективных методов параллельной обработки баз данных в оперативной памяти на современных многопроцессорных вычислительных системах с многоядерными ускорителями, с использованием колоночного представления и сжатия данных. Для решения этой задачи в работах [13, 14] были предложены индексные структуры специального вида, которые называются *распределенными колоночными индексами*. Распределенные колоночные индексы позволяют провести декомпозицию реляционных операций, допускающую их эффективное параллельное выполнение на кластерных вычислительных системах с многоядерными ускорителями. В данной работе рассмотрены вопросы декомпозиции следующих реляционных операций: пересечения, естественного соединения и тета-соединения. Для обозначения реляционных операций в статье используется нотация, заимствованная из монографии [15]. Символом $\circ$ обозначается конкатенация кортежей.

Статья организована следующим образом. В разделе 1 приведено формальное описание колоночного индекса и доменно-интервальной фрагментации, а также доказываются две вспомогательные теоремы. Разделы 2–4 посвящены описаниям методов декомпозиции соответствующих реляционных операций на основе колоночных индексов с доменно-интервальной фрагментацией. Для каждой операции дается формальное описание метода декомпозиции и доказывается теорема, подтверждающая его корректность. В заключении суммируются полученные результаты, делаются итоговые выводы и даются направления дальнейших исследований.

1. Колоночный индекс и доменно-интервальная фрагментация

Под $R\left(A^*, B_1, \ldots, B_u\right)$ будем понимать отношение R с первичным ключом A и атрибутами $B_1, \ldots, B_u$, представляющее собой множество кортежей длины $u+1$ вида $(a, b_1, \ldots, b_u)$, где $a \in \mathbb{Z}_{\geq 0}$ и $\forall j \in \{1, \ldots, u\}\left(b_j \in \mathfrak{D}_{B_j}\right)$. Здесь $\mathfrak{D}_{B_j}$ - домен атрибута B_j. Через $r.B_j$ будем обозначать значение атрибута B_j, через $r.A$ – значение *первичного ключа* в кортеже r: $r = (r.A, r.B_1, \ldots, r.B_u)$. *Первичный ключ* отношения R обладает свойством

$\forall r', r'' \in R\left(r' \neq r'' \Leftrightarrow r'.A \neq r''.A\right)$. Под *адресом кортежа* r мы будем понимать значение первичного ключа этого кортежа. Для получения кортежа отношения R по его адресу будем использовать *функцию разыменования* $\&_R$: $\forall r \in R\left(\&_R(r.A) = r\right)$.

Определение 1. Пусть задано отношение $R\left(A^*, B, \ldots\right)$, $T\left(R\right) = n$. Пусть на множестве $\mathfrak{D}_B$ задано отношение линейного порядка. *Колоночным индексом* $I_{R.B}$ атрибута B отношения R называется упорядоченное отношение $I_{R.B}\left(A^*, B\right)$, удовлетворяющее следующим требованиям:

$$T(I_{R.B}) = n \text{ и } \pi_A\left(I_{R.B}\right) = \pi_A\left(R\right); \tag{1}$$

$$\forall x_1, x_2 \in I_{R.B}\left(x_1 \leq x_2 \Leftrightarrow x_1.B \leq x_2.B\right); \tag{2}$$

$$\forall r \in R\left(\forall x \in I_{R.B}\left(r.A = x.A \Rightarrow r.B = x.B\right)\right). \tag{3}$$

Условие (1) означает, что множества значений первичных ключей (адресов) индекса и индексируемого отношения совпадают. Условие (2) означает, что элементы индекса упорядочены в порядке возрастания значений атрибута B. Условие (3) означает, что атрибут A элемента индекса содержит адрес кортежа отношения R, имеющего такое же значение атрибута B, как и у данного элемента колоночного индекса.

Теорема 1. Пусть задано отношение $R\left(A^*, B, \ldots\right)$. Пусть для отношения R задан колоночный индекс $I_{R.B}$. Тогда

$$\pi_B\left(I_{R.B}\right) = \pi_B\left(R\right). \tag{4}$$

Другими словами, колоночный индекс $I_{R.B}$ представляет все множество значений атрибута B отношения R с учетом повторяющихся значений.

Доказательство. Возьмем произвольное $b \in \mathfrak{D}_B$. Пусть $T\left(\sigma_{B=b}\left(R\right)\right) = k$. Без ограничения общности мы можем считать, что $\forall r \in R\left(r.A < k \Leftrightarrow r.B = b\right)$. Тогда из (1) и (3) следует, что $\forall x \in I_{R.B}\left(x.A < k \Leftrightarrow x.B = b\right)$. Откуда получаем $T\left(\sigma_{B=b}\left(I_{R.B}\right)\right) = k$. Таким образом (4) имеет место. *Теорема доказана.*

Определение 2. Пусть на множестве значений домена $\mathfrak{D}_B$ задано отношение линейного порядка. Пусть также задано разбиение множества $\mathfrak{D}_B$ на $k > 0$ непересекающихся интервалов:

$$\left.\begin{array}{l} V_0 = \left[v_0; v_1\right], V_1 = \left(v_1; v_2\right], \ldots, V_{k-1} = \left(v_{k-1}; v_k\right]; \\ v_0 < v_1 < \ldots < v_k; \\ \mathfrak{D}_B = \bigcup\limits_{i=0}^{k-1} V_i. \end{array}\right\}. \tag{5}$$

Функция $\varphi_{\mathfrak{D}_B} : \mathfrak{D}_B \to \{0, \ldots, k-1\}$ называется *интервальной функцией фрагментации* для домена $\mathfrak{D}_B$, если она удовлетворяет следующему условию:

$$\forall i \in \{0, \ldots, k-1\}\left(\forall b \in \mathfrak{D}_B\left(\varphi_{\mathfrak{D}_B}(b) = i \Leftrightarrow b \in V_i\right)\right). \tag{6}$$

Определение 3. Пусть задан колоночный индекс $I_{R.B}$ для отношения $R\left(A^*,B,\ldots\right)$ с атрибутом B над доменом $\mathfrak{D}_B$ и интервальная функция фрагментации $\varphi_{\mathfrak{D}_B}$. Функция

$$\varphi_{I_{R.B}} : I_{R.B} \to \{0,\ldots,k-1\} \tag{7}$$

называется *доменно-интервальной функцией фрагментации* [3] для индекса $I_{R.B}$, если она удовлетворяет следующему условию:

$$\forall x \in I_{R.B}\left(\varphi_{I_{R.B}}(x) = \varphi_{\mathfrak{D}_B}(x.B)\right). \tag{8}$$

Определим i-тый фрагмент $\left(i = 0,\ldots,k-1\right)$ индекса $I_{R.B}$ следующим образом:

$$I^i_{R.B} = \left\{x \mid x \in I_{R.B};\ \varphi_{I_{R.B}}(x) = i\right\}. \tag{9}$$

Это означает, что в i-тый фрагмент попадают кортежи, у которых значение атрибута B принадлежит i-тому доменному интервалу. Будем называть фрагментацию, построенную таким образом, *доменно-интервальной*. Количество фрагментов k будем называть *степенью фрагментации*.

Доменно-интервальная фрагментация обладает следующими фундаментальными свойствами, вытекающими непосредственно из ее определения:

$$I_{R.B} = \bigcup_{i=0}^{k-1} I^i_{R.B}\ ; \tag{10}$$

$$\forall i,j \in \{0,\ldots,\mathrm{k}-1\}\left(i \neq j \Rightarrow I^i_{R.B} \bigcap I^j_{R.B} = \varnothing\right). \tag{11}$$

Теорема 2. Пусть для колоночного индекса $I_{R.B}$ отношения $R\left(A^*,B,\ldots\right)$ задана доменно-интервальная фрагментация степени k. Тогда

$$\forall i \in \{0,\ldots,\mathrm{k}-1\}\left(\forall x \in I_{R.B}\left(x \in I^i_{R.B} \Leftrightarrow x.B \in V_i\right)\right). \tag{12}$$

Доказательство. Сначала докажем, что

$$\forall i \in \{0,\ldots,\mathrm{k}-1\}\left(\forall x \in I_{R.B}\left(x \in I^i_{R.B} \Rightarrow x.B \in V_i\right)\right). \tag{13}$$

Пусть $x \in I^i_{R.B}$. Тогда из (9) следует $\varphi_{I_{R.B}}(x) = i$. С учетом (8) получаем $\varphi_{\mathfrak{D}_B}(x.B) = i$. Отсюда и из (6) следует $x.B \in V_i$, то есть (13) имеет место. Теперь докажем, что

$$\forall i \in \{0,\ldots,\mathrm{k}-1\}\left(\forall x \in I_{R.B}\left(x.B \in V_i \Rightarrow x \in I^i_{R.B}\right)\right). \tag{14}$$

Пусть $x \in I_{R.B}$ и $x.B \in V_i$. Тогда из (6) следует, что $\varphi_{\mathfrak{D}_B}(x.B) = i$. С учетом (8) получаем $\varphi_{\mathfrak{D}_B}(x.B) = \varphi_{I_{R.B}}(x) = i$. Отсюда и из (9) следует, что $x \in I^i_{R.B}$, то есть (14) имеет место. *Теорема доказана.*

2. Декомпозиция операции пересечения

Пусть заданы два отношения $R\left(A^*,B_1,\ldots,B_u\right)$ и $S\left(A^*,B_1,\ldots,B_u\right)$, имеющие одинаковый набор атрибутов. Мы предполагаем, что $\delta(\pi_{B_1,\ldots,B_u}(R)) = \pi_{B_1,\ldots,B_u}(R)$ и $\delta(\pi_{B_1,\ldots,B_u}(S)) = \pi_{B_1,\ldots,B_u}(S)$, то есть $\pi_{B_1,\ldots,B_u}(R)$ и $\pi_{B_1,\ldots,B_u}(S)$ не содержат дубликатов. Пусть имеется два набора колоночных индексов по атрибутам $B_1,\ldots,B_u$:

$$I_{R.B_1},\ldots,I_{R.B_u};$$
$$I_{S.B_1},\ldots,I_{S.B_u}.$$

Пусть для всех этих индексов задана доменно-интервальная фрагментация степени k :

$$I_{R.B_j}=\bigcup_{i=0}^{k-1}I^i_{R.B_j}; \tag{15}$$

$$I_{S.B_j}=\bigcup_{i=0}^{k-1}I^i_{S.B_j}. \tag{16}$$

Положим

$$P^i_j=\pi_{I^i_{R.B_j}.A\to A_R,\ I^i_{S.B_j}.A\to A_S}\left(I^i_{R.B_j}\underset{(I^i_{R.B_j}.B_j=I^i_{S.B_j}.B_j)}{\bowtie}I^i_{S.B_j}\right) \tag{17}$$

для всех $i=0,\ldots,k-1$ и $j=1,\ldots,u$. Определим

$$P_j=\bigcup_{i=0}^{k-1}P^i_j. \tag{18}$$

Положим

$$P=\bigcap_{j=1}^{u}P_j. \tag{19}$$

Определим

$$Q=\left\{r\mid r\in R\wedge r.A\in\pi_{A_R}(P)\right\}. \tag{20}$$

Теорема 3. $\pi_{B_1,\ldots,B_u}(Q)=\pi_{B_1,\ldots,B_u}(R)\cap\pi_{B_1,\ldots,B_u}(S)$.

Доказательство. Сначала докажем, что

$$\pi_{B_1,\ldots,B_u}(Q)\subset\pi_{B_1,\ldots,B_u}(R)\cap\pi_{B_1,\ldots,B_u}(S). \tag{21}$$

Пусть

$$(a,b_1,\ldots,b_u)\in Q. \tag{22}$$

Из (20) следует, что

$$(a,b_1,\ldots,b_u)=r\in R \tag{23}$$

и

$$a=r.A\in\pi_{A_R}(P). \tag{24}$$

Отсюда следует, что $\exists p\in P\left(p.A_R=a\wedge p.A_S=a'\right)$. С учетом (19) получаем, что $\forall j\in\{1,\ldots,u\}\left(\exists p\in P_j\left(p.A_R=a\wedge p.A_S=a'\right)\right)$. С учетом (18) отсюда получаем $\forall j\in\{1,\ldots,u\}\left(\exists i\in\{0,\ldots,k-1\}\left(\exists p\in P^i_j\left(p.A_R=a\wedge p.A_S=a'\right)\right)\right)$. Отсюда и из (15)-(17) следует, что

$$\forall j\in\{1,\ldots,u\}\left(\exists x\in I_{R.B_j}\left(\exists y\in I_{S.B_j}\left(x.A=a\wedge x.B_j=y.B_j\wedge y.A=a'\right)\right)\right).$$

По определению колоночного индекса отсюда получаем

$$\forall j\in\{1,\ldots,u\}\left(\exists\tilde r\in R\left(\exists\tilde s\in S\left(\tilde r.A=a\wedge\tilde r.B_j=\tilde s.B_j\wedge\tilde s.A=a'\right)\right)\right).$$

Поскольку A является первичным ключом в R и S, с учетом (23) отсюда следует $(a',b_1,\ldots,b_u)\in S$, то есть $(b_1,\ldots,b_u)\in\pi_{B_1,\ldots,B_u}(R)\cap\pi_{B_1,\ldots,B_u}(S)$. Таким образом (21) имеет место. Теперь докажем, что

$$\pi_{B_1,\ldots,B_u}(R)\cap\pi_{B_1,\ldots,B_u}(S)\subset\pi_{B_1,\ldots,B_u}(Q). \tag{25}$$

Пусть

$$(a,b_1,\ldots,b_u)=r\in R \tag{26}$$

и

$$(a',b_1,\ldots,b_u)=s\in S\,. \tag{27}$$

Тогда по определению колоночного индекса с учетом (4) имеем

$$\forall j\in\{1,\ldots,u\}\left(\exists x\in I_{R.B_j}\left(\exists y\in I_{S.B_j}\left(x.A=a\wedge x.B_j=b_j=y.B_j\wedge y.A=a'\right)\right)\right).$$

На основе (12) отсюда получаем

$$\forall j\in\{1,\ldots,u\}\left(\exists i\in\{0,\ldots,\mathrm{k}-1\}\left(\exists x\in I^i_{R.B_j}\left(\exists y\in I^i_{S.B_j}\left(x.A=a\wedge x.B_j=y.B_j\wedge y.A=a'\right)\right)\right)\right).$$

С учетом (17) отсюда следует

$$\forall j\in\{1,\ldots,u\}\left(\exists i\in\{0,\ldots,\mathrm{k}-1\}\left(\exists p\in P^i_j\left(p.A_R=a\wedge p.A_S=a'\right)\right)\right).$$

Применяя (18) получаем

$$\forall j\in\{1,\ldots,u\}\left(\exists p\in P_j\left(p.A_R=a\wedge p.A_S=a'\right)\right).$$

Учитывая (19) отсюда имеем $(a,a')\in P$. Вместе с (20) и (26) это дает

$$(a,b_1,\ldots,b_u)\in Q\,,$$

то есть (25) имеет место. *Теорема доказана.*

3. Декомпозиция операции естественного соединения

Пусть заданы два отношения

$$R\left(A^*,B_1,\ldots,B_u,C_1,\ldots,C_v\right)$$

и

$$S\left(A^*,B_1,\ldots,B_u,D_1,\ldots,D_w\right).$$

Пусть имеется два набора колоночных индексов по атрибутам $B_1,\ldots,B_u$:

$$I_{R.B_1},\ldots,I_{R.B_u}\,;$$

$$I_{S.B_1},\ldots,I_{S.B_u}\,.$$

Пусть для всех этих индексов задана доменно-интервальная фрагментация степени k:

$$I_{R.B_j}=\bigcup_{i=0}^{k-1}I^i_{R.B_j}\,; \tag{28}$$

$$I_{S.B_j}=\bigcup_{i=0}^{k-1}I^i_{S.B_j}\,. \tag{29}$$

Положим

$$P^i_j=\pi_{I^i_{R.B_j}.A\to A_R,\,I^i_{S.B_j}.A\to A_S}\left(I^i_{R.B_j}\underset{I^i_{R.B_j}.B_j=I^i_{S.B_j}.B_j}{\bowtie}I^i_{S.B_j}\right) \tag{30}$$

для всех $i=0,\ldots,k-1$ и $j=1,\ldots,u$. Определим

$$P_j=\bigcup_{i=0}^{k-1}P^i_j\,. \tag{31}$$

Положим

$$P=\bigcap_{j=1}^{u}P_j\,. \tag{32}$$

Определим

$$Q = \{r \circ (s.D_1,\ldots,s.D_w) \mid r \in R \wedge s \in S \wedge (r.A,s.A) \in P\}. \quad (33)$$

Теорема 4. $\pi_{*\setminus A}(Q) = \pi_{*\setminus A}(R) \bowtie \pi_{*\setminus A}(S)$.

Доказательство. Сначала докажем, что

$$\pi_{*\setminus A}(Q) \subset \pi_{*\setminus A}(R) \bowtie \pi_{*\setminus A}(S). \quad (34)$$

Пусть

$$(a,b_1,\ldots,b_u,c_1,\ldots,c_v,d_1,\ldots,d_w) \in Q. \quad (35)$$

Из (33) следует, что существуют кортежи r и s такие, что

$$(a,b_1,\ldots,b_u,c_1,\ldots,c_v) = r \in R, \quad (36)$$

$$(a',b_1',\ldots,b_u',d_1,\ldots,d_w) = s \in S \quad (37)$$

и

$$(r.A,s.A) \in P. \quad (38)$$

Отсюда следует, что $\exists p \in P\left(p.A_R = a \wedge p.A_S = a'\right)$. С учетом (32) получаем, что $\forall j \in \{1,\ldots,u\}\left(\exists p \in P_j\left(p.A_R = a \wedge p.A_S = a'\right)\right)$. С учетом (31) отсюда получаем $\forall j \in \{1,\ldots,u\}\left(\exists i \in \{0,\ldots,k-1\}\left(\exists p \in P_j^i\left(p.A_R = a \wedge p.A_S = a'\right)\right)\right)$. Отсюда и из (28)–(30) следует, что

$$\forall j \in \{1,\ldots,u\}\left(\exists x \in I_{R.B_j}\left(\exists y \in I_{S.B_j}\left(x.A = a \wedge x.B_j = y.B_j \wedge y.A = a'\right)\right)\right).$$

По определению колоночного индекса отсюда получаем

$$\forall j \in \{1,\ldots,u\}\left(\exists \tilde{r} \in R\left(\exists \tilde{s} \in S\left(\tilde{r}.A = a \wedge \tilde{r}.B_j = \tilde{s}.B_j \wedge \tilde{s}.A = a'\right)\right)\right).$$

Поскольку A является первичным ключом в R и S, с учетом (36) и (37) отсюда следует $(a',b_1,\ldots,b_u,d_1,\ldots,d_w) \in S$, то есть $(b_1,\ldots,b_u,c_1,\ldots,c_v,d_1,\ldots,d_w) \in \pi_{*\setminus A}(R) \bowtie \pi_{*\setminus A}(S)$. Таким образом (34) имеет место.

Теперь докажем, что

$$\pi_{*\setminus A}(R) \bowtie \pi_{*\setminus A}(S) \subset \pi_{*\setminus A}(Q). \quad (39)$$

Пусть

$$(a,b_1,\ldots,b_u,c_1,\ldots,c_v) = r \in R \quad (40)$$

и

$$(a',b_1,\ldots,b_u,d_1,\ldots,d_w) = s \in S. \quad (41)$$

Тогда по определению колоночного индекса с учетом (4) имеем

$$\forall j \in \{1,\ldots,u\}\left(\exists x \in I_{R.B_j}\left(\exists y \in I_{S.B_j}\left(x.A = a \wedge x.B_j = b_j = y.B_j \wedge y.A = a'\right)\right)\right).$$

На основе (12) отсюда получаем

$$\forall j \in \{1,\ldots,u\}\left(\exists i \in \{0,\ldots,\mathrm{k}-1\}\left(\exists x \in I_{R.B_j}^i\left(\exists y \in I_{S.B_j}^i\left(x.A = a \wedge x.B_j = y.B_j \wedge y.A = a'\right)\right)\right)\right).$$

С учетом (30) отсюда следует

$$\forall j \in \{1,\ldots,u\}\left(\exists i \in \{0,\ldots,\mathrm{k}-1\}\left(\exists p \in P_j^i\left(p.A_R = a \wedge p.A_S = a'\right)\right)\right).$$

Применяя (31) получаем

$$\forall j \in \{1,\ldots,u\}\left(\exists p \in P_j\left(p.A_R = a \wedge p.A_S = a'\right)\right).$$

Учитывая (32) отсюда имеем $(a,a') \in P$. Вместе с (33), (40) и (41) это дает

$$(a,b_1,\ldots,b_u,c_1,\ldots,c_v,d_1,\ldots,d_w) \in Q,$$

то есть (39) имеет место. *Теорема доказана.*

4. Декомпозиция операции тета-соединения

Пусть заданы два отношения $R\left(A^*,B_1,\ldots,B_u\right)$ и $S\left(A^*,C_1,\ldots,C_v\right)$. Пусть также заданы функции $f:\pi_{B_1,\ldots,B_u}(R)\to\mathbb{R}$ и $g:\pi_{C_1,\ldots,C_v}(S)\to\mathbb{R}$. Определим $\overline{R}=\pi_{A,f(r.B_1,\ldots,r.B_u)\to F}(R)$ и $\overline{S}=\pi_{A,g(s.C_1,\ldots,s.C_v)\to G}(S)$. Пусть имеются колоночные индексы $I_{\overline{R}.F}$ и $I_{\overline{S}.G}$. Пусть для этих индексов задана доменно-интервальная фрагментация степени k :

$$I_{\overline{R}.F}=\bigcup_{i=0}^{k-1}I^i_{\overline{R}.F}\,; \tag{42}$$

$$I_{\overline{S}.G}=\bigcup_{i=0}^{k-1}I^i_{\overline{S}.G}\,. \tag{43}$$

Положим

$$\tilde{P}_i=\pi_{I^i_{\overline{R}.F}.A\to A_R,\ I^i_{\overline{S}.G}.A\to A_S}\left(I^i_{\overline{R}.F}\underset{F<G}{\bowtie}I^i_{\overline{S}.G}\right). \tag{44}$$

Определим

$$M_i=\pi_{A\to A_R}\left(I^i_{\overline{R}.F}\right); \tag{45}$$

$$N_i=\pi_{A\to A_S}\left(I^i_{\overline{S}.G}\right) \tag{46}$$

для всех $i=0,\ldots,k-1$. и

$$\overline{P}_i=M_i\times\bigcup_{l=i+1}^{k-1}N_l\,. \tag{47}$$

Положим

$$P_i=\tilde{P}_i\cup\overline{P}_i; \tag{48}$$

$$P=\bigcup_{i=0}^{k-1}P_i\,. \tag{49}$$

Определим

$$Q=\left\{r\circ s\,|\,r\in R\wedge s\in S\wedge(r.A,s.A)\in P\right\}. \tag{50}$$

Теорема 5. $\pi_{*\setminus R.A,S.A}(Q)=\pi_{*\setminus R.}(R)\underset{f(r.B_1,\ldots,r.B_u)<g(s.C_1,\ldots,s.C_v)}{\bowtie}\pi_{*\setminus A}(S)$.

Доказательство. Сначала докажем, что

$$\pi_{*\setminus A}(Q)\subset\pi_{*\setminus A}(R)\underset{f(r.B_1,\ldots,r.B_u)<g(s.C_1,\ldots,s.C_v)}{\bowtie}\pi_{*\setminus A}(S)\,. \tag{51}$$

Пусть

$$(a,b_1,\ldots,b_u,a',c_1,\ldots,c_v)\in Q\,. \tag{52}$$

Из (50) следует, что существуют кортежи r и s такие, что

$$(a,b_1,\ldots,b_u)=r\in R\,, \tag{53}$$

$$(a',c_1,\ldots,c_v)=s\in S \tag{54}$$

и

$$(r.A,s.A)\in P\,. \tag{55}$$

Отсюда следует, что $\exists p\in P\left(p.A_R=a\wedge p.A_S=a'\right)$. Отсюда и из (49) получаем, что существует $i\in\{0,\ldots,k-1\}$, такой, что

$$p\in P_i\wedge p.A_R=a\wedge p.A_S=a'\,. \tag{56}$$

В силу (48) имеем $p \in \tilde{P}_i \vee p \in \overline{P}_i$. Рассмотрим сначала случай, когда $p \in \tilde{P}_i$. Тогда из (56) и (42)–(44) следует, что

$$\exists x \in I_{\overline{R}.F}\left(\exists y \in I_{\overline{S}.G}\left(x.A = a \wedge x.F < y.G \wedge y.A = a'\right)\right).$$

По определению колоночного индекса отсюда получаем

$$\exists \tilde{r} \in R\left(\exists \tilde{s} \in S\left(\tilde{r}.A = a \wedge \tilde{s}.A = a' \wedge f(\tilde{r}.B_1,\ldots,\tilde{r}.B_u) < g(\tilde{s}.C_1,\ldots,\tilde{s}.C_v)\right)\right). \quad (57)$$

Поскольку A является первичным ключом в R и S, с учетом (53) и (54) отсюда следует, что $\tilde{r} = (a,b_1,\ldots,b_u)$, $\tilde{s} = (a',c_1,\ldots,c_v)$ и $(b_1,\ldots,b_u,c_1,\ldots,c_v) \in \pi_{*\setminus A}(R) \underset{f(r.B_1,\ldots,r.B_u)<g(s.C_1,\ldots,s.C_v)}{\bowtie} \pi_{*\setminus A}(S)$.

Таким образом, (51) имеет место.

Пусть теперь $p \in \overline{P}_i$. Тогда из (56), и (45)–(47) следует, что существуют $i \in \{0,\ldots,k-1\}$ и $l \in \{i+1,\ldots,k-1\}$, такие, что

$$\exists x \in I^i_{\overline{R}.F}\left(x.A = a\right), \quad (58)$$

$$\exists y \in I^l_{\overline{S}.G}\left(y.A = a'\right). \quad (59)$$

Так как $l > i$, то по определению колоночного индекса и доменно-интервальной фрагментации из (58) и (59) получаем, что

$$\exists x \in I_{\overline{R}.F}\left(\exists y \in I_{\overline{S}.G}\left(x.A = a \wedge y.A = a' \wedge x.F < y.G\right)\right),$$

откуда следует (57), и, таким образом, (51) также имеет место.

Теперь докажем, что

$$\pi_{*\setminus A}(R) \underset{\mathrm{f}(r.B_1,\ldots,r.B_u)<\mathrm{g}(s.C_1,\ldots,s.C_v)}{\bowtie} \pi_{*\setminus A}(S) \subset \pi_{*\setminus R.A,S.A}(Q). \quad (60)$$

Пусть

$$(a,b_1,\ldots,b_u) = r \in R, \quad (61)$$

$$(a',c_1,\ldots,c_v) = s \in S, \quad (62)$$

и

$$\mathrm{f}(b_1,\ldots,b_u) < \mathrm{g}(c_1,\ldots,c_v). \quad (63)$$

Тогда по определению колоночного индекса с учетом (4) имеем

$$\exists x \in I_{\overline{R}.F}\left(\exists y \in I_{\overline{S}.G}\left(x.A = a \wedge x.F = \mathrm{f}(b_1,\ldots,b_u) < \mathrm{g}(c_1,\ldots,c_v) = y.G \wedge y.A = a'\right)\right).$$

На основе (5)–(9) и (12) отсюда следует, что возможны только два следующих случая:

1) $\exists i \in \{0,\ldots,\mathrm{k}-1\}\left(x \in I^i_{\overline{R}.F} \wedge y \in I^i_{\overline{S}.G}\right)$; (64)

2) $\exists i \in \{0,\ldots,\mathrm{k}-1\}\left(\exists l \in \{i+1,\ldots,\mathrm{k}-1\}\left(x \in I^i_{\overline{R}.F} \wedge y \in I^l_{\overline{S}.G}\right)\right)$. (65)

Допустим сначала, что имеет место первый случай. Тогда с учетом (44) из (64) следует

$$\exists i \in \{0,\ldots,\mathrm{k}-1\}\left(\exists p \in \tilde{P}_i\left(p.A_R = a \wedge p.A_S = a'\right)\right).$$

Учитывая (48) и (49) отсюда имеем $(a,a') \in P$. Вместе с (50), (61) и (62) это дает

$$(a,b_1,\ldots,b_u,a',c_1,\ldots,c_v) \in Q,$$

то есть (60) имеет место.

Теперь допустим, что имеет место второй случай. Тогда с учетом (47) из (65) следует

$$\exists i \in \{0,\ldots,\mathrm{k}-1\}\left(\exists p \in \overline{P}_i\left(p.A_R = a \wedge p.A_S = a'\right)\right).$$

Учитывая (48) и (49) отсюда имеем $(a,a') \in P$. Вместе с (50), (61) и (62) это дает

$$(a,b_1,\ldots,b_u,a',c_1,\ldots,c_v) \in Q,$$

то есть (60) имеет место. *Теорема доказана.*

Заключение

В статье были представлены формальные описания методов декомпозиции операций пересечения, естественного соединения и тета-соединения по неравенству на основе доменно-интервальной фрагментации колоночных индексов. Для каждого метода была доказана его корректность. Практическая ценность предложенных методов декомпозиции заключается в том, что ресурсоемкие вычисления могут производиться независимо над соответствующими фрагментами. Для операции пересечения это – вычисление P_j^i по формуле (17), для операции естественного это – вычисление P_j^i по формуле (30), для тета-соединения это – вычисление $\tilde{P}_i$ по формуле (44). В дальнейшем мы планируем разработать методы декомпозиции для операций тета-соединения с конъюнкцией, дизъюнкцией и отрицанием, а также для операции группировки.

Работа выполнена при финансовой поддержке Минобрнауки РФ в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014—2020 годы» (Госконтракт № 14.574.21.0035).

Литература

1. Turner, V. The Digital Universe of Opportunities: Rich Data and the creasing Value of the Internet of Things. — White paper. — International Data Corporation. — 2014. / V. Turner, J.F. Gantz, D. Reinsel, et al. URL: http://idcdocserv.com/1678 (дата обращения: 29.01.2015)
2. Соколинский, Л.Б. Параллельные машины баз данных / Л.Б. Соколинский // Природа. — 2001. — № 8. — С. 10–17.
3. Соколинский, Л.Б. Параллельные системы баз данных / Л.Б. Соколинский — Москва: Издательство Московского государственного университета, 2013. — 184 с.
4. Sokolinsky, L.B. Design and Evaluation of Database Multiprocessor Architecture with High Data Availability / L.B. Sokolinsky // Proceedings of the 12th International workshop on database and expert systems applications. — IEEE Computer Society, 2001. — P. 115–120.
5. Pan, C.S. Taming Elephants, or How to Embed Parallelism into PostgreSQL / C.S. Pan, M.L. Zymbler // Lecture Notes in Computer Science. — 2013. — Vol. 8055, Part 1. — P. 153–164.
6. Костенецкий, П.С. Моделирование иерархических многопроцессорных систем баз данных / П.С. Костенецкий, Л.Б. Соколинский // Программирование. — 2013. — Т. 39, № 1. — С. 3–22.
7. Plattner, H. In-Memory Data Management: An Inflection Point for Enterprise Applications / H. Plattner, A. Zeier — Springer, 2011. — 254 p.
8. Abadi, D.J. Column-Stores vs. Row-Stores: How Different Are They Really? / D.J. Abadi, S.R. Madden, N. Hachem // Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 9–12, 2008, Vancouver, BC, Canada. — ACM, 2008. — P. 967–980.
9. Fang, J. Sesame: A User-Transparent Optimizing Framework for Many-Core Processors / J. Fang, A.L. Varbanescu, H. Sips // Proceedings of the 13th IEEE/ACM International

Symposium on Cluster, Cloud and Grid Computing (CCGrid2013), May 13–16, 2013, Delft, Netherlands. — IEEE, 2013. — P. 70–73.

10. Breß, S. Efficient Co-Processor Utilization in Database Query Processing / S. Breß, F. Beier, H. Rauhe, et al. // Information Systems. — 2013. — Vol. 38, No. 8. — P. 1084–1096.
11. Scherger, M. Design of an In-Memory Database Engine Using Intel Xeon Phi Coprocessors / M. Scherger // Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'14), July 21–24, 2014, Las Vegas, USA. — CSREA Press, 2014. — P. 21–27.
12. Беседин, К.Ю. Моделирование обработки запросов на гибридных вычислительных системах с многоядерными сопроцессорами и графическими ускорителями / К.Ю. Беседин, П.С. Костенецкий // Программные системы: теория и приложения. — 2014. — Т. 5, № 1-1 (19). — С. 91–110.
13. Иванова, Е.В. Использование распределенных колоночных индексов для выполнения запросов к сверхбольшим базам данных / Е.В. Иванова, Л.Б. Соколинский // Параллельные вычислительные технологии (ПАВТ'2014). Труды международной научной конференции. — Челябинск: Издательский центр ЮУрГУ, 2014. — С. 270–275.
14. Иванова, Е.В. Использование распределенных колоночных хеш-индексов для обработки запросов к сверхбольшим базам данных / Е.В. Иванова // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров. Труды Международной суперкомпьютерной конференции. — М.: Изд-во МГУ, 2014. — С. 102–104.
15. Гарсиа-Молина, Г. Системы баз данных. Полный курс. / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом — М.: Издательский дом «Вильямс». — 2004. — 1088 с.

Иванова Елена Владимировна, программист отдела поддержки и обучения пользователей, Лаборатория суперкомпьютерного моделирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), ivanovaev@susu.ac.ru.

Соколинский Леонид Борисович, проректор по информатизации, Южно-Уральский государственный университет (Челябинск, Российская Федерация), Leonid.Sokolinsky@susu.ru.

Поступила в редакцию 4 февраля 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 1, pp. 44–56

DOI: 10.14529/cmse150104

DECOMPOSITION OF INTERSECTION AND JOIN OPERATIONS BASED ON THE DOMAIN-INTERVAL FRAGMENTED COLUMN INDICES

E.V. Ivanova, South Ural State University (Chelyabinsk, Russian Federation),
L.B. Sokolinsky, South Ural State University (Chelyabinsk, Russian Federation)

The paper presents decomposition of relational operations based on distributed column indices and domain-interval fragmentation. This decomposition admits parallel executing the resource-intensive relational operations without data transfers. All column index fragments are stored in main memory in compressed form to conserve space. During the parallel execution of relational operations, compressed index fragments are loaded on different processor cores. These cores uncompress fragments, perform relational operations and compress fragments of partial result, which is a set of keys. Partial results are merged in the resulting set of keys. DBMS use the resulting set of keys for building the resulting table. Described approach allows efficient parallel query processing for very large databases on modern computing cluster systems with many-core accelerators.

Keywords: very large databases, parallel query processing, column indices, domain-interval fragmentation, decomposition of relational operations.

References

1. Turner V., Gantz J.F, Reinsel D., et al. The Digital Universe of Opportunities: Rich Data and the creasing Value of the Internet of Things. White paper. International Data Corporation. 2014. URL: http://idcdocserv.com/1678 (accessed: 29.01.2015)
2. Sokolinsky L.B. Parallel'nye mashiny baz dannyh [Parallel Database Machines]. Priroda [Nature]. 2001. No 8. P. 10–17.
3. Sokolinsky L.B. Parallel'nye systemy baz dannyh [Parallel Database Systems]. Moscow, Publishing of the Moscow State University, 2013. 184 p.
4. Sokolinsky L.B. Design and Evaluation of Database Multiprocessor Architecture with High Data Availability // Proceedings of the 12th International workshop on database and expert systems applications. IEEE Computer Society, 2001. P. 115–120.
5. Pan C.S., Zymbler M.L. Taming Elephants, or How to Embed Parallelism into PostgreSQL // Lecture Notes in Computer Science. 2013. Vol. 8055, Part 1. P. 153–164.
6. Kostenetsky P.S., Sokolinsky L.B. Modelirovanie ierarhicheskih mnogoprocessornyh sistem baz dannyh [Simulation of Hierarchical Multiprocessor Database Systems]. Programmirovanie [Programming]. 2013. Vol. 39, No 1. P. 3–22.
7. Plattner H., Zeier A. In-Memory Data Management: An Inflection Point for Enterprise Applications Springer, 2011. 254 p.
8. Abadi D.J., Madden S.R, Hachem N. Column-Stores vs. Row-Stores: How Different Are They Really? // Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 9–12, 2008, Vancouver, BC, Canada. ACM, 2008. P. 967–980.

9. Fang J., Varbanescu A.L., Sips H. Sesame: A User-Transparent Optimizing Framework for Many-Core Processors // Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2013), May 13–16, 2013, Delft, Netherlands. IEEE, 2013. P. 70–73.
10. Breß S., Rauhe H, et al. Efficient Co-Processor Utilization in Database Query Processing // Information Systems. 2013. Vol. 38, No. 8. P. 1084–1096.
11. Scherger M. Design of an In-Memory Database Engine Using Intel Xeon Phi Coprocessors // Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'14), July 21–24, 2014, Las Vegas, USA. CSREA Press, 2014. P. 21–27.
12. Besedin K.Y., Kostenetsky P.S. Modelirovanie obrabotki zaprosov na gibridnyh vychislitel'nyh sistemah s mnogojadernymi soprocessorami i graficheskimi uskoriteljami [Simulation of Query Processing on Hybrid Computing Systems with Multi-core Coprocessors and Graphic Accelerators]. Programmnye sistemy: teorija i prilozhenija [Software Systems: Theory and Applications]. 2014. Vol. 5, No 1-1 (19). P. 91–110.
13. Ivanova E.V., Sokolinsky L.B. Ispol'zovanie raspredelennykh kolonochnykh indeksov dlya vypolneniya zaprosov k sverkhbol'shim bazam dannykh [Using Distributed Column Indeces for Query Execution for Very Large Databases]. Parallel'nye vychislitel'nye tekhnologii (PaVT'2014). Trudy mezhdunarodnoy nauchnoy konferentsii [Proceedings of the International Conference Parallel Computational Technologies (PCT'2014)]. Chelyabinsk: SUSU publishing center. 2014. P. 270–275.
14. Ivanova E.V. Ispol'zovanie raspredelennyh kolonochnyh hesh-indeksov dlja obrabotki zaprosov k sverhbol'shim bazam dannyh [Using Distributed Column Hash Indices for the Query for Very Large Databases] // Nauchnyj servis v seti Internet: mnogoobrazie superkomp'juternyh mirov. Trudy Mezhdunarodnoy nauchnoy konferentsii [Proceedings of the International Scientific Conference Scientific Service on the Internet: the Variety of Supercomputing Worlds]. Moscow. Bulletin of publishing house of the Moscow university. 2014. P. 102–104.
15. Garcia-Molina H., Ullman J., Widom J. Database Systems: The Complete Book. Prentice Hall Press, 2008. 1248 p.

Received February 4, 2015.

УДК 004.75 DOI: 10.14529/cmse150105

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЙ ВИРТУАЛЬНЫЙ СКРИНИНГ В ENTERPRISE DESKTOP GRID НА БАЗЕ BOINC[1]

Е.Е. Ивашко, Н.Н. Никитина, С. Меллер

В работе представлен опыт разработки и использования распределенной вычислительной инфраструктуры Enterprise Desktop Grid на базе программной платформы BOINC для проведения виртуального скрининга с использованием свободно распространяемого программного обеспечения и открытых баз данных моделей химических соединений. Согласно концепции Enterprise Desktop Grid, в рамках виртуальной единой вычислительной сети объединяются неспециализированные вычислители, принадлежащие одной организации или группе пользователей, напрямую заинтересованных в решении прикладной задачи. В работе приводится описание и оценка производительности Enterprise Desktop Grid, созданной в рамках совместного научно-исследовательского проекта Института экспериментальной дерматологии при университете г. Любек (Германия) и Института прикладных математических исследований Карельского научного центра РАН. Приведены выводы о применимости подхода, а также краткий обзор полученных результатов виртуального скрининга.

Ключевые слова: Desktop Grid, Enterprise Desktop Grid, распределенные вычисления, виртуальный скрининг.

Введение

В биологии и медицине в настоящее время выделяется ряд актуальных научно-исследовательских задач, решение которых требует значительных вычислительных ресурсов. Многие из этих задач обусловлены острой необходимостью в разработке новых лекарственных средств (часть из которых может стать настоящим прорывом в лечении редких или новых заболеваний) и повышении эффективности и безопасности клинически испытанных препаратов. Этапы испытания лекарств в лаборатории и клинических условиях предваряет сложный процесс отбора и подготовки «кандидатов». В ходе этого процесса используются различные методы как на этапе выбора макромолекулы-«мишени», играющей ключевую роль в протекании заболевания, так и на этапе подбора химического соединения (лиганда), которое должно связываться с молекулой-мишенью, подавляя или усиливая ее биологические функции.

Одним из этапов разработки лекарства является виртуальный скрининг, подразумевающий обработку больших баз данных моделей химических соединений с целью подбора наиболее подходящего для мишени лиганда. Как правило, оценка степени взаимодействия определенного лиганда с молекулой-мишенью проводится на основе специальных математических моделей и не требует большого количества вычислительных ресурсов. Однако отбор пяти-восьми наиболее подходящих лигандов из нескольких десятков миллионов кандидатов требует привлечения методов и средств высокопроизводительных вычислений. При этом оценка каждого лиганда-кандидата проводится независимо от остальных, что позволяет разбить исходную задачу на миллионы не связанных между собой подзадач.

В работе представлен опыт проведения виртуального скрининга с помощью вычислительных средств Enterprise Desktop Grid на базе программной платформы BOINC. Произво-

[1]Статья рекомендована к публикации программным комитетом суперкомпьютерной конференции «Научный сервис в сети Интернет — 2014».

дительность созданной вычислительной инфраструктуры проиллюстрирована результатами, полученными в ходе выполнения реального исследовательского проекта, проводимого совместно сотрудниками Института экспериментальной дерматологии (ранее — Клиники дерматологии, аллергологии и венерологии) при университете г. Любек (Германия) и Института прикладных математических исследований Карельского научного центра РАН.

Работа организована следующим образом. В разделе 1 описана прикладная задача виртуального скрининга лекарств. Раздел 2 посвящен одному из способов решения данной задачи — использованию распределенной вычислительной системы типа Enterprise Desktop Grid на базе BOINC. В заключении приводится краткий обзор результатов, полученных в работе.

1. Виртуальный скрининг

Методы, используемые для подбора лигандов, принято делить на две группы: синтез принципиально новых молекул с желаемыми свойствами и подбор лигандов на основе структурных свойств мишени. Во втором случае исследование требует перебора больших баз данных химических соединений, поэтому не может быть целиком основано на физических экспериментах. На данном этапе целесообразно провести отбор потенциальных кандидатов при помощи молекулярного докинга — компьютерного моделирования процесса связывания лиганда с мишенью в трехмерном пространстве на основе специальных математических и физических моделей. Процесс компьютерного отбора лигандов для заданной мишени называется виртуальным скринингом [11]. Его эффективность для получения биологически значимых подмножеств лигандов доказывают успешные результаты целого ряда проектов [3, 7, 10].

В качестве входных данных для виртуального скрининга используются пространственные модели молекул белка и лигандов. В процессе молекулярного докинга подбирается взаимное геометрическое расположение молекул, минимизирующее свободную энергию Гиббса для данной молекулярной системы. Затем вычисляется значение так называемой оценочной функции, выражающее силу связывания при выбранном расположении молекул. Процесс повторяется для заданного множества лигандов; для этапа физических экспериментов выбираются те лиганды, для которых была предсказана наилучшая сила связывания.

Существует целый ряд баз данных лигандов, в том числе, доступные в открытом доступе [8]. Примерами открытых баз данных моделей химических соединений, подготовленных для виртуального скрининга, являются ZINC [6] (более 35 млн. записей), ChemSpider [4] (более 30 млн. записей), CoCoCo [5] (порядка 7 млн. записей). В связи с большим объемом обрабатываемых баз данных и сложностью трехмерных моделей, проведение виртуального скрининга требует значительного количества вычислительных ресурсов. Помимо больших баз данных структурных моделей белков и лигандов, в открытом доступе также находится апробированное в ходе выполнения крупных проектов открытое программное обеспечение для молекулярного докинга.

Общую продолжительность виртуального скрининга можно проиллюстрировать на примере открытого ПО AutoDock Vina [9]. Данное ПО разработано для молекулярного докинга белковых молекул и лигандов, и по результатам специализированных тестов находится на одном уровне или превосходит по скорости и точности целый ряд эффективных программных решений, включая коммерческие.

Расчет энергии связывания 1 млн. лигандов с использованием AutoDock Vina с минимальной точностью на современном четырехъядерном персональном компьютере занимает порядка 1800 час. процессорного времени, и время расчетов растет с увеличением требуемой точности. Таким образом, проведение виртуального скрининга, например, по БД ZINC для единственной модели белка на современном настольном компьютере потребовало бы не менее 8,5 лет.

2. Enterprise Desktop Grid на базе BOINC

Проведение виртуального скрининга состоит из нескольких миллионов не связанных между собой подзадач. Эта особенность позволяет использовать не дорогостоящие мощности вычислительных кластеров, а более доступные и масштабируемые средства распределенных вычислений. В частности, ряд научно-исследовательских проектов (наиболее масштабный из которых — Docking@Home) использовали для виртуального скрининга системы Desktop Grid, основанные на принципах добровольных вычислений (volunteer computing).

Desktop Grid — это объединение в качестве единого логического «суперкомпьютера» большого количества неспециализированных вычислителей (офисных рабочих и персональных компьютеров, ноутбуков) относительно невысокой производительности. Основные достоинства технологии при соответствующей реализации — практически неограниченная масштабируемость и, следовательно, пиковая производительность, устойчивость к сбоям, минимальная стоимость создания и сопровождения. Однако такие системы подходят для решения только тех задач, которые могут быть разбиты на независимые и менее ресурсоемкие подзадачи.

Стандартом де-факто при организации добровольных вычислений является платформа BOINC (Berkeley Open Infrastructure for Network Computing) [2]. BOINC отличается простотой в установке, настройке и администрировании, обладает хорошими возможностями по масштабируемости, обеспечивает простое подключение вычислительных узлов и использование дополнительного ПО, дает возможности интеграции с другими грид-системами и др. Платформа имеет архитектуру «клиент–сервер», при этом клиентская часть может работать на компьютерах с различными аппаратными и программными характеристиками. Платформа BOINC была реализована для вычислений, необходимых для анализа космических радиосигналов, в рамках проекта SETI@Home на добровольно предоставляемых персональных компьютерах частных лиц и организаций, объединенных сетью Интернет. С момента создания на основе BOINC было реализовано множество проектов добровольных вычислений для исследований в различных отраслях наук, и на сегодняшний день суммарная пиковая мощность задействованных в вычислениях ресурсов составляет порядка 7,71 петафлопс.

Организация и сопровождение широкомасштабного проекта добровольных вычислений (наподобие Docking@Home) требуют большого внимания и сил для взаимодействия с сообществом и обеспечения функционирования проекта. Такой подход может оказаться нецелесообразным, если эксперименты проводятся нерегулярно, с промежутками на обработку результатов. Тогда более выгодной оказывается концепция Enterprise Desktop Grid, где в рамках виртуальной единой вычислительной сети собираются неспециализированные вычислители, принадлежащие одной организации или группе пользователей, напрямую заинтересованных в получении результатов экспериментов.

Enterprise Desktop Grid по сравнению с Desktop Grid, использующимися при проведении добровольных вычислений, имеют ряд преимуществ: вычислительные узлы являются надежными, что снижает уровень репликации и кворума; вычислительным процессом можно управлять централизованно; высокая скорость передачи данных по локальной сети и др. [1]. Возможности BOINC (репликация, кворумы, версионность приложений и разнообразие платформ) позволяют эффективно задействовать вычислительные ресурсы Enterprise Desktop Grid.

Для проведения виртуального скрининга в рамках совместного научно-исследовательского проекта Клиники дерматологии университета г. Любек (Германия) и Института прикладных математических исследований Карельского научного центра РАН была построена Enterprise Desktop Grid на базе BOINC. В представленной работе проводилось исследование свойств одного из белков в рамках научной работы Института экспериментальной дерматологии университета г. Любек. Для виртуального скрининга была задана пространственная модель белка и несколько целевых подмножеств лигандов из базы данных ZINC.

К концу июля 2014 г. в состав Enterprise Desktop Grid вошли 52 вычислительных узла — настольные персональные компьютеры (ПК) сотрудников Института экспериментальной дерматологии, ресурсы вычислительного кластера (незадействованные другими задачами), серверы и ПК сотрудников Института прикладных математических исследований Карельского научного центра РАН. Пиковая вычислительная мощность системы (по оценкам BOINC) составила 4801 гигафлопс. Такое количество вычислительных узлов обеспечило комфортный анализ поступающих результатов. На первом этапе экспериментов был проведен виртуальный скрининг по 8 279 803 лигандам с использованием ПО AutoDock Vina, было отобрано более 17 тыс. лигандов для дальнейших экспериментов, 41 лекарственное средство рекомендовано для тестирования в лабораторных условиях. Общая продолжительность виртуального скрининга составила 27 недель, а суммарное процессорное время расчетов — 31,7 лет. В настоящее время на Enterprise Desktop Grid продолжаются расчеты нескольких десятков тысяч лигандов из целевых множеств.

На текущий момент полученные результаты позволяют говорить об успешности проекта. Отобранные лиганды-кандидаты пройдут экспериментальную проверку в лаборатории университета г. Любек.

Заключение

В работе представлены результаты разработки и апробации системы Enterprise Desktop Grid для проведения виртуального скрининга одного из белков. Для организации вычислений использована программная платформа BOINC, в качестве вычислительных узлов задействованы компьютеры Института экспериментальной дерматологии университета г. Любек (Германия) и Института прикладных математических исследований Карельского научного центра РАН. Подход показал применимость, позволив в краткие сроки организовать вычислительную сеть, и обеспечив исследователей необходимыми результатами моделирования.

На первом этапе виртуального скрининга получен ряд биологически значимых результатов и оценена вычислительная эффективность системы. В дальнейшем планируется продолжение экспериментов на основе данных, полученных из лаборатории универси-

тета г. Любек, а также разработка и реализация алгоритмов повышения эффективности Enterprise Desktop Grid в задачах виртуального скрининга.

Работа выполнена при финансовой поддержке РФФИ (проект 13-07-00008), Программы стратегического развития Петрозаводского государственного университета и Германской службы академических обменов DAAD.

Литература

1. Ивашко, Е.Е. Desktop Grid корпоративного уровня / Е.Е. Ивашко — Программные системы: теория и приложения. — 2014. — No. 1(19). — С. 183–190.
2. Anderson, D.P. BOINC: A system for public-resource computing and storage / D.P. Anderson // R. Buyya (Editor), Fifth IEEE/ACM International Workshop on Grid Computing. — 2004. — P. 4–10.
3. Breakthrough in the fight against childhood cancer. URL: `http://www.worldcommunitygrid.org/about_us/viewNewsArticle.do?articleId=342` (дата обращения: 14.11.2014).
4. ChemSpider: Search and share chemistry. URL: `http://www.chemspider.com` (дата обращения: 14.11.2014).
5. Del Rio, A. CoCoCo: a free suite of multiconformational chemical databases for highthroughput virtual screening purposes / A. Del Rio, A.J. Moura Barbosa, F. Caporuscio, G.F. Mangiatordi — Molecular BioSystems. — 2010. — No. 6. — P. 2122–2128.
6. Irwin, J.J. ZINC: A Free Tool to Discover Chemistry for Biology / J.J. Irwin, T. Sterling, M.M. Mysinger, et al. — Journal of Chemical Information and Modeling. — 2012. — No. 52(7). — P. 1757–1768.
7. Matter, H. Applications and Success Stories in Virtual Screening / H. Matter, C. Sotriffer — Virtual Screening: Principles, Challenges, and Practical Guidelines (ed. C. Sotriffer). Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, 2011. — P. 319–358.
8. Moura Barbosa, A.J. Freely accessible databases of commercial compounds for highthroughput virtual screenings / A.J. Moura Barbosa, A. Del Rio — Current Topics in Medicinal Chemistry. — 2012. — No. 12. — P. 866–877.
9. Trott, O. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading / O. Trott, A.J. Olson — Journal of Computational Chemistry. — 2010. — No. 31. — P. 455–461.
10. Villoutreix, B.O. Structure-based virtual ligand screening: recent success stories / B.O. Villoutreix, R. Eudes, M.A. Miteva — Combinatorial chemistry & high throughput screening. — 2009. — No. 12(10). — P. 1000–1016.
11. Walters, W.P. Virtual screening an overview / W.P. Walters, M.T. Stahl, M.A. Murcko — Drug Discovery Today. — 1998. — Vol. 3, No. 4. — P. 160–178.

Ивашко Евгений Евгеньевич, к.ф.-м.н., научный сотрудник Лаборатории телекоммуникационных систем, Институт прикладных математических исследований Карельского научного центра РАН (Петрозаводск, Российская Федерация), ivashko@krc.karelia.ru.

Никитина Наталия Николаевна, стажер-исследователь Лаборатории информационных компьютерных технологий, Институт прикладных математических исследований Карельского научного центра РАН (Петрозаводск, Российская Федерация), nikitina@krc.karelia.ru.

Dr. Steffen Möller, PhD, research associate of the University of Lübeck, The Lübeck Institute of Experimental Dermatology (Lübeck, Germany), steffen.moeller@uksh.de.

Поступила в редакцию 19 ноября 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 1, pp. 57–63

DOI: 10.14529/cmse150105

HIGH-PERFORMANCE VIRTUAL SCREENING IN A BOINC-BASED ENTERPRISE DESKTOP GRID

E.E. Ivashko, Institute of Applied Mathematical Research of the Karelian Research Centre of the Russian Academy of Sciences (Petrozavodsk, Russia)
ivashko@krc.karelia.ru,
N.N. Nikitina, Institute of Applied Mathematical Research of the Karelian Research Centre of the Russian Academy of Sciences (Petrozavodsk, Russia)
nikitina@krc.karelia.ru,
S. Möller, University of Lübeck, The Lübeck Institute of Experimental Dermatology (Lübeck, Germany) steffen.moeller@uksh.de

In the paper we present the experience of developing and using an Enterprise Desktop Grid infrastructure based on BOINC software platform and designed for virtual screening using open-source software and open databases of chemical compounds models. According to the idea of Enterprise Desktop Grid, a virtual single computational network integrates non-specialized computers within an organization or group of users that are directly interested in solving an applied problem. We describe and evaluate the performance of the infrastructure created withing a joint research project between the Lübeck Institute of Experimental Dermatology (former Clinic of dermatology, allergology and venerology) of the University of Lübeck and the Institute of Applied Mathematical Research of the Karelian Research Centre of the Russian Academy of Sciences. We summarize the project and briefly overview the results of the virtual screening.

Keywords: Desktop Grid, Enterprise Desktop Grid, distributed computing, virtual screening.

References

1. Ivashko E.E. Enterprise Desktop Grids // Programmnye Sistemy: Teoriya i Prilozheniya [Program Systems: Theory and Applications]. 2014. No. 1(19). P. 83–190.
2. Anderson D.P. BOINC: A system for public-resource computing and storage // R. Buyya (Editor), Fifth IEEE/ACM International Workshop on Grid Computing. 2004. P. 4–10.
3. Breakthrough in the fight against childhood cancer. URL: http://www.worldcommunitygrid.org/about_us/viewNewsArticle.do?articleId=342 (accessed: 14.11.2014).
4. ChemSpider: Search and share chemistry. URL: http://www.chemspider.com (accessed: 14.11.2014).

5. Del Rio A., Moura Barbosa A.J., Caporuscio F., Mangiatordi G.F. CoCoCo: a free suite of multiconformational chemical databases for high-throughput virtual screening purposes // Molecular BioSystems. 2010. No. 6. P. 2122–2128.
6. Irwin J.J., Sterling T., Mysinger M.M., et al. ZINC: A Free Tool to Discover Chemistry for Biology // Journal of Chemical Information and Modeling. 2012. No. 52(7). P. 1757–1768.
7. Matter H., Sotriffe C. Applications and Success Stories in Virtual Screening // Virtual Screening: Principles, Challenges, and Practical Guidelines (ed. C. Sotriffer). Wiley-VCH Verlag GmbH & Co. KGaA, Weinheim, Germany, 2011. P. 319–358.
8. Moura Barbosa A.J., Del Rio A. Freely accessible databases of commercial compounds for high-throughput virtual screenings // Current Topics in Medicinal Chemistry. 2012. No. 12. P. 866–877.
9. Trott O., Olson A.J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading // Journal of Computational Chemistry. 2010. No. 31. P. 455–461.
10. Villoutreix B.O., Eudes R., Miteva M.A. Structure-based virtual ligand screening: recent success stories // Combinatorial Chemistry & High Throughput Screening. 2009. No. 12(10). P. 1000–1016.
11. Walters W.P., Stahl M.T., Murcko M.A. Virtual screening an overview // Drug Discovery Today. 1998. Vol. 3, No. 4. P. 160–178.

Received November 19, 2014.

УДК 004.65 DOI: 10.14529/cmse150106

ПРИМЕНЕНИЕ ГРАФИЧЕСКИХ УСКОРИТЕЛЕЙ ДЛЯ ОБРАБОТКИ ЗАПРОСОВ НАД СЖАТЫМИ ДАННЫМИ В ПАРАЛЛЕЛЬНЫХ СИСТЕМАХ БАЗ ДАННЫХ[1]

С.О. Приказчиков, П.С. Костенецкий

Работа посвящена вопросам применения графических процессоров для обработки запросов в параллельных системах баз данных. Целью данной работы является оценка эффективности выполнения запросов к сжатой базе данных без предварительной распаковки с использованием графических ускорителей, поддерживающих технологию CUDA. Объем внутренней памяти ГПУ на порядки меньше, чем объем оперативной памяти современных вычислительных систем. Это ограничивает размер базы данных, которую можно загрузить в память ГПУ и как следствие не позволяет раскрыть весь вычислительный потенциал графического процессора. Предлагается подход для обработки запросов над сжатыми данными на ГПУ. На основе предложенного подхода реализован эмулятор параллельной СУБД. Аналогичный эмулятор разработан для ЦПУ. Приведены результаты вычислительных экспериментов и произведена оценка эффективности данного подхода.

Ключевые слова: графические процессоры, параллельная обработка запросов, CUDA.

Введение

В настоящее время известно большое количество исследований обработки запросов к базам данных в оперативной памяти [4, 5, 9, 12], но использование графических ускорителей (ГПУ) и многоядерных сопроцессоров [11] открывает новые перспективы исследований в этой области [1, 2, 7]. Целью данной работы является оценка эффективности выполнения запросов к сжатой базе данных без предварительной распаковки с использованием графических ускорителей, поддерживающих технологию CUDA. Объем внутренней памяти ГПУ на порядки меньше, чем объем оперативной памяти современных вычислительных систем. Это ограничивает размер базы данных, которую можно загрузить в память ГПУ и как следствие не позволяет раскрыть весь вычислительный потенциал ГПУ. Для снижения влияния объема видеопамяти на возможность хранения в ней больших баз данных применяется сжатие. Предполагается, что сжатие позволит не только увеличить объем хранимой базы данных, но и повысит производительность по сравнению с обработкой несжатых данных.

Статья организована следующим образом. В разделе 1 описана реализация алгоритма выборки для ГПУ и ЦПУ. Раздел 2 описывает проводимые вычислительные эксперименты. В заключении подведены итоги исследования.

[1] Статья рекомендована к публикации программным комитетом Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: многообразие суперкомпьютерных миров – 2014»

1. Реализация

На текущий момент разработан эмулятор СУБД [6], позволяющий выполнять запрос SELECT над сжатыми данными, непосредственно на ГПУ с использованием технологии CUDA и, для сравнения, эмулятор для ЦПУ, работающий по технологии OpenMP.

Для сжатия был выбран и реализован алгоритм RLE [8, 10]. Основным преимуществом использования данного алгоритма в СУБД является возможность выполнения многих реляционных запросов без предварительной распаковки данных.

Реляционное отношение базы данных хранится в сжатом виде в памяти ГПУ. Кортежи отношения имеют по два атрибута, первый атрибут — номер кортежа (идентификатор), второй атрибут — значение (строка символов). На первом этапе алгоритм производит поиск строк, длина которых совпадает с размером искомой строки. Так, каждый блок производит выборку из нескольких подряд идущих кортежей. Каждый поток блока выполняет сравнение одного символа искомой строки с символом в строке базы данных. При такой организации распараллеливания соблюдается принцип memory coalescing [3], что позволяет свести к минимуму влияние латентности доступа к памяти ГПУ. Для распараллеливания на ГПУ используется разбиение кортежей реляционного отношения по блокам CUDA. На конечном этапе номера кортежей, содержащих совпавшие строки, записываются в выходной массив. Передаются не кортежи целиком, а только их номера с целью уменьшить объем передаваемых данных по шине PCI Express. Шина PCI Express имеет на несколько порядков меньшую скорость передачи данных, чем оперативная память и внутренняя память ГПУ, поэтому данный подход позволяет значительно повысить общую производительность при обработке данных.

2. Вычислительные эксперименты

Вычислительные эксперименты проводились с использованием разработанного эмулятора СУБД на оборудовании, характеристики которого приведены в табл. 1.

Таблица 1

Оборудование для экспериментов

Оборудование	Характеристики	
Процессор	Intel Core i7-3610QM 2.3 ГГц	
ОЗУ	8 Гб DDR3-1600	
Твердотельный накопитель	Plextor M5 Pro PX-128M5P, 128 Гб, SATA III	
Системная шина	PCI Express 2.0	
Графический ускоритель	Модель	NVIDIA GeForce GTX 670M
	Объем видеопамяти	3 072 Мб
	Ядер CUDA	336
	Тактовая частота ядра	620 МГц
	Тактовая частота памяти	3 000 МГц
	Пропускная способность памяти	72 Гб/с
	Разрядность памяти	192-bit

Для проведения вычислительных экспериментов было сгенерировано 4 тестовых отношения. Размер отношений, а также коэффициент сжатия строк в кортежах представлены в табл. 2.

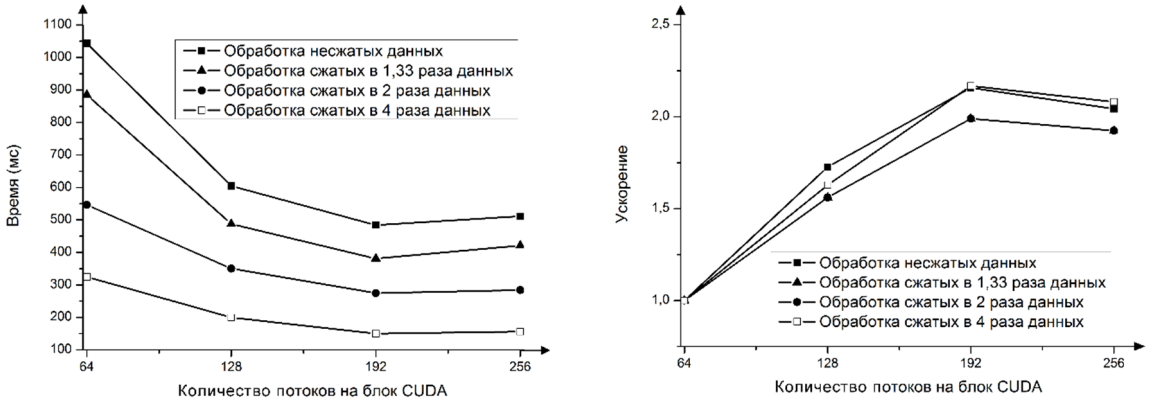
Таблица 2

Характеристики тестовых отношений

№ тестовой БД	Размер отношения (количество кортежей)	Коэффициент сжатия	Объем в сжатом виде, Мб
1	33 000 000	4	519
2		2	1 039
3		1,33	1 562
4		1	2 077

2.1. Исследование эффективности операции выборки над сжатыми данными

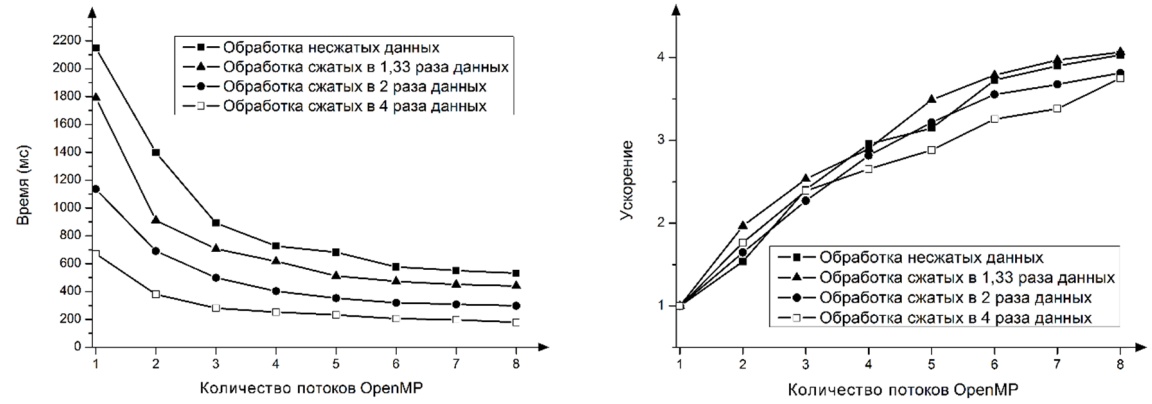
Было произведено исследование эффективности операции выборки над сжатыми данными в сравнении с обработкой несжатых данных.



а) Время **б)** Ускорение

Рис. 1. Выполнение алгоритма ГПУ SELECT на одном вычислительном узле

Как видно из графиков времени выполнения операции выборки из тестовых баз данных с использованием ГПУ (см. рис. 1а) и с использованием ЦПУ (см. рис. 2а), при применении сжатия мы получаем максимальный прирост скорости в 3,2 раза на ГПУ и в 3 раза на ЦПУ.



а) Время **б)** Ускорение

Рис. 2. Выполнение алгоритма ЦПУ SELECT на одном вычислительном узле

На графике сравнения времени выполнения операции выборки на ГПУ и на ЦПУ (см. рис. 5) видно, что при использовании ГПУ выборка выполняется быстрее независимо от размера базы данных.

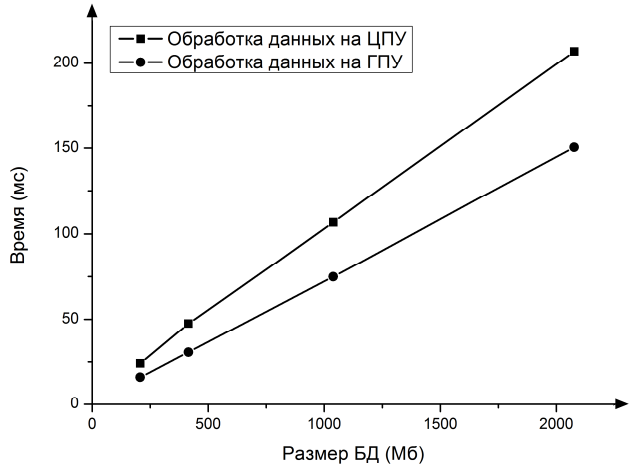


Рис. 3. Время выполнения алгоритмов ЦПУ и ГПУ SELECT на одном вычислительном узле над сжатыми в 4 раза данными

Заключение

В работе рассматривались вопросы применения графических процессоров для обработки запросов в параллельных системах баз данных. Предложен подход для обработки запросов над сжатыми данными на ГПУ. На основе предложенного подхода реализован эмулятор параллельной СУБД. Аналогичный эмулятор разработан для ЦПУ. Приведены результаты вычислительных экспериментов и произведена оценка эффективности данного подхода. Установлено, что выполнение выборки над сжатыми данными на ГПУ эффективнее, чем на ЦПУ в 1,2 раза. Небольшое преимущество ГПУ объясняется тем, что в качестве тестовой конфигурации имелся достаточно производительный процессор Intel Core i7-3610QM и не очень производительный графический ускоритель NVIDIA GeForce GTX 670M. На следующем этапе исследования тестирование будет произведено на новейших ГПУ NVIDIA Tesla K40 и многоядерных сопроцессорах Intel Xeon Phi.

Работа выполнена при финансовой поддержке Минобрнауки РФ в рамках ФЦП «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014–2020 годы» (Соглашение № 14.574.21.0035).

Литература

1. Беседин, К.Ю. Моделирование обработки запросов на гибридных вычислительных системах с многоядерными сопроцессорами и графическими ускорителями / К.Ю. Беседин, П.С. Костенецкий // Программные системы: теория и приложения: электрон. научн. журн. Института программных систем им. А.К. Айламазяна РАН. — 2014. — Т. 5, № 1(19). — С. 91–110.
2. Беседин, К.Ю. Применение многоядерных сопроцессоров в параллельных системах баз данных / К.Ю. Беседин, П.С. Костенецкий // Параллельные вычислительные технологии (ПаВТ'2013): труды международной научной конференции (1–5 апреля 2013 г., г. Челябинск). — Челябинск: Издательский центр ЮУрГУ, 2013. — С. 583.

3. Боресков, А.В. Параллельные вычисления на GPU. Архитектура и программная модель CUDA / А.В. Боресков, Н.Д. Марковский, Д.Н. Микушин и др. — М.: Издательство Московского университета, 2012. — 336 с.
4. Костенецкий, П.С. Технологии параллельных систем баз данных для иерархических многопроцессорных сред / П.С. Костенецкий, А.В. Лепихов, Л.Б. Соколинский // Автоматика и телемеханика. — 2007. — № 5. — С. 112–125.
5. Костенецкий, П.С. Моделирование иерархических многопроцессорных систем баз данных / П.С. Костенецкий, Л.Б. Соколинский // Программирование. – Москва: МАИК «Наука/Интерпериодика». — 2013. — Т. 39, № 1. — С. 3–22.
6. Костенецкий, П.С. Моделирование параллельных систем баз данных: учебное пособие / П.С. Костенецкий, Л.Б. Соколинский — Челябинск: Фотохудожник, 2012. — 78 с.
7. Сафина, Ю.Н. Моделирование аппаратной архитектуры и коммуникационных сетей вычислительных кластеров с гибридными узлами для параллельных систем баз данных / Ю.Н. Сафина, П.С. Костенецкий // Параллельные вычислительные технологии (ПаВТ'2012): Труды международной научной конференции (г. Новосибирск, 26–30 марта 2012 г.). — Челябинск: Издательский центр ЮУрГУ, 2012. — С. 741.
8. Abadi, D. Integrating compression and execution in column-oriented database systems / D. Abadi, S. Madden, M. Ferreira // Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, 2006. — P. 671–682.
9. Пан, К.С. Разработка параллельной СУБД на основе последовательной СУБД PostgreSQL с открытым исходным кодом / К.С. Пан, М.Л. Цымблер // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». — 2012. — № 18(277). Вып. 12. — С. 112–120.
10. Костенецкий, П.С. Исследование эффективности различных методов сжатия при передаче данных из основной памяти в память сопроцессора Intel Xeon Phi / П.С. Костенецкий, К.Ю. Беседин // Вычислительные методы и программирование. — 2014. — Т. 15, № 4. — С. 593–601.
11. Иванова, Е.В. Использование распределенных колоночных индексов для выполнения запросов к сверхбольшим базам данных / Е.В. Иванова, Л.Б. Соколинский // Параллельные вычислительные технологии (ПаВТ'2014): труды международной научной конференции (1–3 апреля 2014 г., г. Ростов-на-Дону). — Челябинск: Издательский центр ЮУрГУ, 2014. — С. 270–275.
12. Соколинский, Л.Б. Параллельные машины баз данных / Л.Б. Соколинский // Природа. — 2001. — № 8. — С. 10–17.

Приказчиков Степан Олегович, старший лаборант лаборатории суперкомпьютерного моделирования, Южно-Уральский государственный университет (Челябинск, Россия), prikazchikovso@susu.ru.

Костенецкий Павел Сергеевич, к.ф.-м.н., доцент, руководитель лаборатории суперкомпьютерного моделирования, Южно-Уральский государственный университет (Челябинск, Россия), kostenetskiy@susu.ru.

Поступила в редакцию 11 декабря 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 1, pp. 64–70

DOI: 10.14529/cmse150106

USING GRAPHICS ACCELERATORS FOR QUERY PROCESSING OVER COMPRESSED DATA IN PARALLEL DATABASE SYSTEMS

S.O. Prikazchikov, South Ural State University (Chelyabinsk, Russian Federation)
prikazchikovso@susu.ru,
P.S. Kostenetskiy, South Ural State University (Chelyabinsk, Russian Federation)
kostenetskiy@susu.ru.

This article talks about using graphics processors for query processing in parallel database systems. The goal is to evaluate query execution efficiency over compressed database without decompression on multicore GPUs which support CUDA technology. GPU's memory size is significantly smaller than modern computer system's RAM size. This fact affects database's size can be loaded into GPU's internal memory, thus computing potential of GPU can not be used efficiently. The new approach presented in this article allows query processing over compressed data on GPU. An emulator of parallel DBMS is developed based on this approach. The similar emulator for a CPU is designed. Results of computational experiments are presented and analysis of efficiency of the proposed approaches is performed.

Keywords: databases, graphics processors, parallel query processing.

References

1. Besedin K.Y, Kostenetskiy P.S. Simulating of query processing on multiprocessor database systems with modern coprocessors // MIPRO 2014 Proceedings of the 37th International Convention. May 26–30, 2014. Opatija. Croatia, 2014. P. 1835–1837.
2. Besedin K.Y., Kostenetskiy P.S. Primenenie mnogoyadernykh soprotsessorov v parallel'nykh sistemakh baz dannykh [Using multi-core coprocessors in parallel database systems]. Parallel'nye vychislitel'nye tekhnologii (PAVT'2013): Trudy mezhdunarodnoy nauchnoy konferentsii (1–5 aprelya 2013 g., g. Chelyabinsk). Chelyabinsk: Izdatel'skiy tsentr YuUrGU [Proceedings of the International Conference Parallel Computational Technologies (PCT'2014), April 1–5, 2013, Chelyabinsk, Russia. Chelyabinsk: SUSU publishing center], 2013. P. 583.
3. Boreskov A.V, Markovskiy N.D., Mikushin D.N. et al. Parallel'nye vychisleniya na GPU. Arkhitektura i programmnaya model' CUDA [Parallel calculations on GPU. Architecture and program CUDA model]. M.: Izdatel'stvo Moskovskogo universiteta [Bulletin of publishing house of the Moscow university], 2012. 58 p.

4. Kostenetskii P.S., Lepikhov A.V., Sokolinskii L.B. Technologies of parallel database systems for hierarchical multiprocessor environments // Automation and Remote Control. 2007. Vol. 68, No. 5. P. 847–859.
5. Kostenetskii P.S., Sokolinsky L.B. Simulation of Hierarchical Multiprocessor Database Systems // Programming and Computer Software. Vol. 39, No. 1. 2013. P. 10–24.
6. Kostenetskiy P.S., Sokolinskiy L.B. Modelirovanie parallel'nykh sistem baz dannykh: uchebnoe posobie [Modeling of parallel database systems: manual]. Chelyabinsk: Fotokhudozhnik [Bulletin Chelyabinsk: Pictorialist], 2012. 78 p.
7. Safina Yu.N., Kostenetskiy P.S. Modelirovanie apparatnoy arkhitektury i kommunikatsionnykh setey vychislitel'nykh klasterov s gibridnymi uzlami dlya parallel'nykh sistem baz dannykh [Simulation of hardware architecture and communication networks of computing clusters with hybrid nodes for parallel database systems].Parallel'nye vychislitel'nye tekhnologii (PaVT'2012): Trudy mezhdunarodnoy nauchnoy konferentsii (g. Novosibirsk, 26–30 marta 2012 g.) Chelyabinsk: Izdatel'skiy tsentr YuUrGU [Proceedings of the International Conference Parallel Computational Technologies (PCT'2012), March 26–30, 2012, Novosibirsk, Russia. Chelyabinsk: SUSU publishing center]. 2012. P. 741.
8. D. Abadi, S. Madden, M. Ferreira. Integrating compression and execution in column-oriented database systems // Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, 2006. P. 671–682.
9. Pan C.S., Zymbler M.L. Razrabotka parallelnoj SUBD na osnove posledovatelnoj SUBD PostgreSQL s otkrytym ishodnym kodom [Development of a Parallel Database Management System on the Basis of Open-Source PostgreSQL DBMS]. Vestnik Yuzho-Uralskogo gosudarstvennogo universiteta. Seriya «Matematicheskoe modelirovanie i programmirovanie» [Bulletin of South Ural State University. Series: Mathematical Modeling, Programming & Computer Software]. 2012. No. 18(277). Vol. 12. P. 112–120.
10. Kostenetskiy P.S., Besedin K.Yu. Issledovanie effektivnosti razlichnykh metodov szhatiya pri peredache dannykh iz osnovnoy pamyati v pamyat' soprotsessora Intel Xeon Phi [Efficiency Evaluation of Some Compression Methods for Data Transfer between Main Memory and Intel Xeon Phi Coprocessors]. Vychislitel'nye metody i programmirovanie [Numerical methods and programming]. 2014. Vol. 15, No. 4. P. 593–601.
11. Ivanova E.V., Sokolinsky L.B. Ispol'zovanie raspredelennykh kolonochnykh indeksov dlya vypolneniya zaprosov k sverkhbol'shim bazam dannykh [Using distributed column indexes for query execution over very large databases]. Parallel'nye vychislitel'nye tekhnologii (PaVT'2014): trudy mezhdunarodnoy nauchnoy konferentsii (1–3 aprelya 2014 g., g. Rostov-na-Donu). Chelyabinsk: Izdatel'skiy tsentr YuUrGU [Proceedings of the International Conference Parallel Computational Technologies (PCT'2014), April 1–3, 2014, Rostov-na-Donu, Russia. Chelyabinsk: SUSU publishing center]. 2014. P. 270–275.
12. Sokolinsky L.B. Parallel'nye mashiny baz dannyh [Parallel database machines]. Priroda [Nature]. 2001. № 8. P. 10–17.

Received December 11, 2014.

УДК 519.61+512.552.7 DOI: 10.14529/cmse150107

ВЫЧИСЛЕНИЕ РАНГОВ ГРУПП ЦЕНТРАЛЬНЫХ ЕДИНИЦ ЦЕЛОЧИСЛЕННЫХ ГРУППОВЫХ КОЛЕЦ КОНЕЧНЫХ ГРУПП

Р.Ж. Алеев, Н.А. Цыбина

Изучение центральных единиц (центральных обратимых элементов) целочисленных групповых колец конечных групп почти всегда приводит к трудоемким вычислениям, как в случае нахождения отдельных центральных единиц, так и при описании групп центральных единиц. В силу того, что периодическая часть групп тривиальна (с точностью до знака это элементы центра группы), более интересно нахождение сведений о части без кручения, которая является прямым произведением бесконечных циклических групп. Число таких бесконечных прямых сомножителей — ранг группы центральных единиц. Поэтому ранги групп центральных единиц целочисленных групповых колец конечных групп — одна из важнейших характеристик таких групп. Поэтому вычисление рангов групп центральных единиц представляет большой интерес при изучении групп центральных единиц. В работе приведены формулы для вычисления рангов в общем случае и в нескольких важнейших частных случаях. На основании этих формул произведены вычисления рангов в достаточно широких диапазонах. Для вычислений использовалась система компьютерной алгебры GAP. Результаты вычислений показываются в таблицах и на графике.

Ключевые слова: характер группы, центральная единица, ранг абелевой группы, система GAP.

Введение

В России (и ранее в СССР) работы по вычислительным аспектам алгебры и теории чисел не часты. Авторы не могут надеяться, что данная статья заполнит образовавшуюся лакуну, но могут надеяться, что возникнет интерес к подобным направлениям исследований.

В 2014 г. исполнилось ровно 25 лет, как первый из авторов начал заниматься центральными единицами целочисленных групповых колец. Почти с самого начала стало ясно, что это направление исследований часто приводит к таким вычислениям, которые невозможно проделать вручную, и приходилось их выполнять на компьютерах. В дальнейшем неоднократно производились различные вычисления по данной тематике, некоторые из которых вошли, как существенная часть в [2].

Далее под *центральной единицей* будет всегда (если не оговорено противное) пониматься *центральная единица (центральный обратимый элемент)* целочисленного группового кольца конечной группы. Среди задач, связанных с центральными единицами выделим задачу об нахождении рангов групп центральных единиц. Напомним, что под *рангом* конечнопорожденной абелевой группы понимается число бесконечных прямых сомножителей при разложении группы в прямое произведение циклических подгрупп. Группы центральных единиц конечнопорождены, поэтому ранг является важной необходимой характеристикой группы центральных единиц, так как периодическая часть всегда тривиальна (равна $\langle -1 \rangle \times Z(G)$) [10].

В этой работе будем рассматривать вопросы, связанные с вычислениями рангов групп центральных единиц. Статья организована следующим образом. В первом разделе приводятся формулы для вычисления рангов групп центральных единиц, как общие так и для

отдельных классов конечных групп. Во втором разделе приводятся результаты вычислений рангов в виде таблиц и графика и их анализ. Следует отметить, что невозможно привести полные результаты вычислений в связи с их экстремально большим объемами. В заключении приведена краткая сводка полученных результатов и указаны направления будущих исследований.

1. Формулы для вычисления рангов групп центральных единиц

Обозначения. *Будем придерживаться следующих обозначений.*

1) *Зафиксируем конечную группу* G.
2) *Пусть* χ — *характер группы* G. *Тогда:*
   а) $\deg\chi$ — *его степень,*
   б) $\mathbb{Q}(\chi)$ — *поле характера* χ,
   в) $d_\chi = [\mathbb{Q}(\chi):\mathbb{Q}]$ — *степень* $\mathbb{Q}(\chi)$ *над* $\mathbb{Q}$,
   г) I_χ — *множество всех характеров алгебраически сопряженных с* χ.
3) I_G — *множество представителей классов алгебраически сопряженных неприводимых характеров.*
4) $I_{\mathbb{R}} = \{\chi \in I_G \mid \mathbb{Q}(\chi) \subset \mathbb{R}\}$.
5) $I_{\mathbb{C}} = \{\chi \in I_G \mid \mathbb{Q}(\chi) \not\subseteq \mathbb{R}\}$.

Отсюда

$$I_G = I_{\mathbb{R}} \cup I_{\mathbb{C}},\ I_{\mathbb{R}} \cap I_{\mathbb{C}} = \varnothing.$$

Следующий результат получен в [10].

Лемма 1. При введенных выше обозначениях для конечной группы G:

1) группа единиц кольца целых центра рациональной групповой алгебры изоморфна прямому произведению групп единиц колец целых полей $\mathbb{Q}(\chi)$ для всех $\chi \in I_G$;
2) группа центральных единиц целочисленного группового кольца и группа единиц кольца целых центра рациональной групповой алгебры имеют одинаковые ранги.

Как следствие, из этой леммы легко получается следующий результат.

Лемма 2. [формула для рангов] Ранг центральных единиц целочисленного группового кольца равен

$$\sum_{\chi\in I_{\mathbb{R}}} d_\chi + \frac{1}{2}\sum_{\chi\in I_{\mathbb{C}}} d_\chi - |I_G|.$$

Это утверждение приведено в [2, с. 152, следствие 3.1] и нигде не опубликовано, поскольку считалось тривиальным.

На основе этой формулы были проведены вычисления рангов для всех групп, представленных в GAP [14] таблицами характеров (на 2000 г.), и это было в [2, Приложение Г.1, с. 323–336], но нигде не анонсировалось и не публиковалось.

1) Число таблиц характеров — 998.
2) Полученные значения рангов (записаны списком в виде пар, где первый элемент — значение ранга, а второй — число групп с данным значением ранга):

```
[ [ 0, 336 ], [ 1, 135 ], [ 2, 108 ], [ 3, 69 ],  [ 4, 65 ],
  [ 5, 40 ], [ 6, 42 ], [ 7, 20 ], [ 8, 27 ],  [ 9, 20 ],
```

```
[ 10, 11 ], [ 11, 14 ], [ 12, 4 ],  [ 13, 12 ], [ 14, 11 ],
[ 15, 11 ], [ 16, 8 ],  [ 17, 2 ], [ 18, 3 ], [ 19, 10 ],
[ 20, 7 ], [ 21, 1 ],  [ 22, 1 ], [ 23, 3 ], [ 24, 1 ],
[ 25, 1 ], [ 27, 6 ],  [ 28, 2 ], [ 29, 2 ], [ 31, 1 ],
[ 32, 4 ], [ 34, 2 ],  [ 35, 3 ], [ 36, 2 ], [ 40, 3 ],
[ 42, 2 ], [ 43, 1 ],  [ 45, 1 ], [ 59, 1 ], [ 69, 2 ],
[ 79, 1 ], [ 94, 1 ],  [ 126, 1 ], [ 146, 1 ] ]
```

Позднее были найдены (с помощью В.Д. Мазурова) статьи [11] и [13], в которых были другие формулы для рангов.

Лемма 3. Пусть G — конечная группа и r — число классов сопряженности группы G (равносильно число неприводимых комплексных характеров). Тогда

$$r = \sum_{\chi \in I_G} d_\chi.$$

Доказательство. Так как сопряженных с χ точно d_χ, то $r = \sum_{\chi \in I_G} d_\chi$. □

Определение 1. Элементы $a, b \in G$ называются $\mathbb{Q}$-*сопряженными,* если существует такой $x \in G$, что $x^{-1}bx = a^s$, где $(s, |G|) = 1$.

Определение 2. $\mathbb{Q}$-*класс* с представителем a назовем множество $\{a^G\}_\mathbb{Q}$ всех $\mathbb{Q}$-сопряженных с a, то есть

$$\{a^G\}_\mathbb{Q} = \bigcup_{s=1(s,|G|)=1}^{|a|-1} \left\{(a^s)^G\right\}.$$

Обозначение. *Число $\mathbb{Q}$-классов группы G обозначим через $n_\mathbb{Q}$.*

Лемма 4. Пусть G — конечная группа. Тогда $n_\mathbb{Q} = |I_G|$.

Доказательство. Так каждая орбита при действии $\mathrm{Gal}(\mathbb{Q}(\chi))$ дает одну простую компоненту $Z(\mathbb{Q}G)$, то из теоремы Бермана–Витта [8, с. 265, 287] сразу следует $n_\mathbb{Q} = |I_G|$. □

Определение 3. Элементы $a, b \in G$ называются $\mathbb{R}$-*сопряженными*, если существует такой $x \in G$, что $x^{-1}bx = a^{\pm 1}$.

Определение 4. $\mathbb{R}$-*класс* с представителем a назовем множество $\{a^G\}_\mathbb{R}$ всех $\mathbb{R}$-сопряженных с a, то есть

$$\{a^G\}_\mathbb{R} = \{a^G\} \cup \{(a^{-1})^G\}.$$

Обозначение. *Число $\mathbb{R}$-классов группы G обозначим через $n_\mathbb{R}$.*

Определение 5. Класс сопряженности a^G называется *действительным = вещественным* классом, если $a^{-1} \in \{a^G\}$.

Обозначение. *Число действительных классов группы G обозначим через $h_\mathbb{R}$.*

Определение 6. Характер χ называется *действительным = вещественным*, если поле характера $\mathbb{Q}(\chi) \subset \mathbb{R}$.

Лемма 5.

1) $h_{\mathbb{R}} = \sum_{\chi \in I_{\mathbb{R}}} d_\chi$,
2) если c число классов сопряженности $\{a^G\}$ с условием $\{a^G\} \neq \{(a^{-1})^G\}$, то

$$\begin{cases} r = h_{\mathbb{R}} + 2c \\ n_{\mathbb{R}} = h_{\mathbb{R}} + c \end{cases}.$$

Доказательство. Так как сопряженных с χ точно d_χ, то $h_{\mathbb{R}} = \sum_{\chi \in I_{\mathbb{R}}} d_\chi$.

Используя лемму 3 и первое утверждение, получим второе. □

Обозначение. *Обозначим через $r_{\mathbb{Z}}$ ранг группы центральных единиц целочисленного группового кольца группы G.*

Теперь по лемме 2

$$r_{\mathbb{Z}} = \sum_{\chi \in I_{\mathbb{R}}} d_\chi + \frac{1}{2} \sum_{\chi \in I_{\mathbb{C}}} d_\chi - |I_G|.$$

Отсюда, очевидно, имеем:

$$r_{\mathbb{Z}} = \frac{1}{2} \sum_{\chi \in I_G} d_\chi + \frac{1}{2} \sum_{\chi \in I_{\mathbb{R}}} d_\chi - |I_G|.$$

Подставляя в приведенную выше формулу значения

$$\sum_{\chi \in I_G} d_\chi,\ \sum_{\chi \in I_{\mathbb{R}}} d_\chi,\ |I_G|$$

из лемм 3, 4 и 5, мы получим

$$r_{\mathbb{Z}} = \frac{1}{2} r + \frac{1}{2} h_{\mathbb{R}} - n_{\mathbb{Q}} = \frac{1}{2} \left(r + h_{\mathbb{R}} - 2n_{\mathbb{Q}} \right).$$

Это и есть формула Риттера и Сегала из [13].

Из системы в лемме 5 имеем

$$c = r - n_{\mathbb{R}},\ h_{\mathbb{R}} = n_{\mathbb{R}} - c = n_{\mathbb{R}} - r + n_{\mathbb{R}} = 2n_{\mathbb{R}} - r.$$

Подставим выражение для $h_{\mathbb{R}}$ в формулу Риттера и Сегала и получим

$$r_{\mathbb{Z}} = \frac{1}{2} \left(r + 2n_{\mathbb{R}} - r - 2n_{\mathbb{Q}} \right) = n_{\mathbb{R}} - n_{\mathbb{Q}}.$$

Это и есть формула Ферраза [11].

Замечание 1. Таким образом, формулы из [11] и [13] достаточно просто выводятся из леммы 2.

Подведем итог в виде теоремы.

Теорема. *Пусть G — конечная группа. При сохранении введенных ранее обозначений ранг группы центральных единиц целочисленного группового кольца группы G может вычислен по одной из следующих формул:*

1) $\displaystyle\sum_{\chi\in I_{\mathbb{R}}} d_\chi + \frac{1}{2}\sum_{\chi\in I_{\mathbb{C}}} d_\chi - |I_G| = \frac{1}{2}\sum_{\chi\in I_G} d_\chi + \frac{1}{2}\sum_{\chi\in I_{\mathbb{R}}} d_\chi - |I_G|,$ (формула из [2])

2) $n_{\mathbb{R}} - n_{\mathbb{Q}},$ (формула из [11])

3) $\dfrac{1}{2}\left(r + h_{\mathbb{R}} - 2n_{\mathbb{Q}}\right).$ (формула из [13])

Приведем, как следствия, формулы для рангов для некоторых классов групп. Однако следует отметить, что в действительности следствия 1–5 каждый раз получались без применения каких-либо общих формул. Исключение составляет последнее следствие 6, полученное по формуле из [11].

Обозначение. *Для натурального числа n пусть $\tau(n)$ — количество его натуральных делителей.*

Следствие 1. [Алеев, [1]] Ранг группы центральных единиц целочисленного группового кольца группы $L_2(2^n)$ равен

$$2^n + 1 - \tau(2^n - 1) - \tau(2^n + 1).$$

Следствие 2. [Алеев—Ишечкина, [3]] Пусть $q = 2^{2n+1}$, $n \geqslant 1$, $m_+ = q{+}1{+}2r$, $m_- = q{+}1{-}2r$, где $r = 2^n = \sqrt{\frac{q}{2}}$. Тогда ранг группы центральных единиц целочисленного группового кольца группы $Sz(q)$ равен

$$q + 2 - \tau(m_+) - \tau(m_-) - \tau(q - 1).$$

Следствие 3. [Алеев—Митина, [4]] Ранг группы центральных единиц целочисленного группового кольца группы $PGL_2(q)$, q нечетно, равен

$$q + 2 - \tau(q - 1) - \tau(q + 1).$$

Следствие 4. [Алеев—Перавина, [5]] Ранг группы центральных единиц целочисленного группового кольца группы $L_2(q)$, q нечетно, равен

$$\frac{q+3}{2} - \tau\left(\frac{q-1}{2}\right) - \tau\left(\frac{q+1}{2}\right) + 1,$$

при $q \equiv 1 \pmod 4$ и q — не квадрат,

$$\frac{q+3}{2} - \tau\left(\frac{q-1}{2}\right) - \tau\left(\frac{q+1}{2}\right),$$

в остальных случаях.

Следствие 5. [Алеев, [10]] Пусть G — циклическая группа порядка n. Тогда ранг группы центральных целочисленного группового кольца группы G равен

$$\left[\frac{n}{2}\right] - \tau(n) + 1 = \begin{cases} \frac{n+1}{2} - \tau(n) & \text{для нечетного } n, \\ \frac{n}{2} - \tau(n) + 1 & \text{для четного } n. \end{cases}$$

Следствие 6. [Шумакова, [9]] Пусть G — метациклическая группа Фробениуса с циклическим ядром порядка m и циклическим дополнением порядка n (n делит n). Тогда ранг группы центральных единиц целочисленного группового кольца группы G равен

$$\left(1+\left[\frac{n}{2}\right]-\frac{n}{2}\right)\frac{m-1}{n}-\left[\frac{n}{2}\right]+2-\tau(m)-\tau(n)=$$
$$=\begin{cases}\frac{m-1}{2n}+\frac{n-1}{2}+2-\tau(m)-\tau(n) & \text{для нечетного } n,\\ \frac{m-1}{n}+\frac{n}{2}+2-\tau(m)-\tau(n)+1 & \text{для четного } n.\end{cases}$$

2. Вычисления рангов

В этом разделе приведем результаты вычислений рангов для случаев, описанных в следствиях 1–6. Вычисления производились в системе GAP [12] с привлечением дополнительных средств для анализа полученных результатов.

Во всех следствиях 1–6 есть «неинтересная» часть, вычисляемая легко (как правило, это линейная функция) по параметрам группы (например, для следствия 3 это $q+2$), и «интересная» часть, связанная с нахождением количества делителей некоторых чисел. Интерес ко второй части побуждается тем, что количество делителей числа напрямую не связано с данным числом. А именно, если $n=p_1^{\alpha_1}\cdots p_k^{\alpha_k}$ — разложение числа в произведение степеней различных простых чисел (каноническое разложение), то количество делителей

$$\tau(n)=(\alpha_1+1)\cdots(\alpha_k+1).$$

Однако следует отметить, что этот подход приводит к задаче факторизации натуральных чисел, которая наряду с задачей дискретного логарифма, является (как общепризнано) очень трудной[1].

Соглашение. *Мы сосредоточимся на вычислениях, связанных с количеством делителей.*

2.1. Степени 2

В этом пункте мы рассмотрим случаи для групп $L_2(2^n)$ и $Sz(2^{2n+1})$. Так как согласно леммам 1 и 2 вычисления зависят от 2^n, то вполне понятно, что здесь можно вычислить небольшое число возможностей для n.

2.1.1. Группы $L_2(2^n)$

Согласно следствию 1 нужно вычислить сумму $s(n)=\tau(2^n-1)+\tau(2^n+1)$ для натуральных n, что было проделано для n от 1 до 102. Приведем код программы при интерактивном вводе в GAPe на рис. 1.

Последнее сообщение в программе

```
gap> Tau(2^103+1);
#I  FactorsInt: used the following factor(s) which are probably primes:
#I        8142767081771726171
```

означает, что появляется трудность с факторизацией $2^{103}+1$.

Табл. 1 показывает зависимость между n и $s(n)$.

[1]Трудность задач факторизации и дискретного логарифма широко используется для целей защиты информации, например, [6] и [7]

```
# Задание списка
gap> t:=[];
[  ]
# Заполнение списка
# Tau - встроенная функция GAP для подсчета числа делителей
gap> for n in [1..102] do
> i:=2^n;
> tm1:=Tau(i-1);
> tp1:=Tau(i+1);
> Add(t,[tm1+tp1,n]);
> od;
# Сортировка
gap> Sort(t);
gap> for i in [1..Length(t)] do
# Запись списка в файл
> AppendTo("Tau2p.txt",t[i],"\n");
> od;
# Возникновение трудности для 103
gap> Tau(2^103+1);
#I  FactorsInt: used the following factor(s) which are probably primes:
#I        8142767081771726171
gap> quit;
```

Рис. 1. Нахождение $s(n)$

Таблица 1

Зависимость между n и $s(n)$

$s(n)$	n	$s(n)$	n	$s(n)$	n	$s(n)$	n	$s(n)$	n
3	1	24	21, 22, 25, 26, 34, 38, 85	88	87	196	40	960	78
4	2	28	12, 27, 39	96	91, 98	208	42	1040	88
5	3	32	93	104	24	224	54, 75	2048	102
6	4, 5, 7, 13, 17, 19, 31, 61	36	32, 97	112	55	256	81	4624	60
8	11, 23, 101	40	33, 57, 62, 69	128	45	264	56	5632	90
10	6, 8	48	18, 35, 46	132	64	448	66	6152	96
12	9, 37, 41, 43, 49, 67, 79	52	20	136	44, 52	512	99	8284	72
14	10	64	74, 82, 86, 95	140	68	516	92	8256	100
16	14, 29, 47, 53, 71, 73	68	28, 83	144	30, 76	528	36	9248	84
18	16, 18	72	58, 65	160	63	608	70		
20	15, 59	80	51, 77, 94	192	50	776	48, 80		

2.1.2. Группы $Sz(2^{2n+1})$

По следствию 2 нужно вычислить сумму

$$s_1(n) = \tau(2^{2n+1} - 2^{n+1} + 1) + \tau(2^{2n+1} + 2^{n+1} + 1) + \tau(2^{2n+1} + 1)$$

для натуральных n, что было проделано для таких n от 1 до 75, а при $n > 75$ возникают трудности с факторизацией.

На рис. 2 изображены точки с координатами $(2n + 1, s_1(n))$, и становится ясно, что распределение этих точек весьма хаотично.

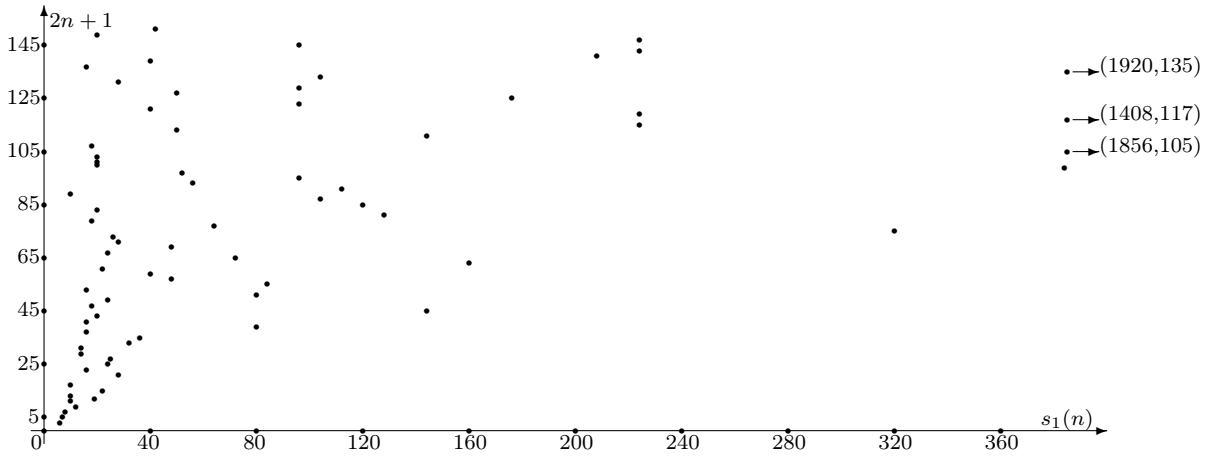


Рис. 2. Распределение $2n + 1$ и $s_1(n)$

Табл. 2 показывает зависимость между n и $s_1(n)$.

Таблица 2

Зависимость между n и $s_1(n)$

$s_1(n)$	$2n+1$	$s_1(n)$	$2n+1$	$s_1(n)$	$2n+1$	$s_1(n)$	$2n+1$
6	3	24	25, 27, 49, 67	56	93	160	63
7	5	26	73	64	77	208	141
8	7	28	21, 71, 131	80	39, 51	224	115, 119, 143, 147
10	11, 13, 17, 89	32	33	84	55	320	75
12	9, 19	36	35	96	95, 129, 145	384	99
14	29, 31	40	59, 121, 139	104	87, 133	1408	117
16	23,37,41, 53,137	42	151	112	91	1856	105
18	47, 79, 107	48	57, 79	120	85	1920	135
20	43,83,101, 103,109,149	50	113	128	81		
22	15, 61	52	97	144	45, 111		

2.2. Группы $PGL_2(q)$ и $PSL_2(q)$, q нечетно

Для построения таблиц в этом пункте 2.2. были найдены все нечетные примарные q, от 3 до $2^{26}+1 = 67\,108\,865$. Отметим, что таких q в указанном диапазоне будет 3 958 973.

2.2.1. Группы $PGL_2(q)$, q нечетно

По следствию 3 нужно вычислить сумму $s_2(q) = \tau(q-1) + \tau(q+1)$.

Таблица 3

Все значения $s_2(q)$

$3 \leqslant s_2(q) \leqslant 140$	$5, 7, 8, [10..14], [16..18], [20..118], [120..124], 126, 128, [129..134], [136..140],$
$142 \leqslant s_2(q) \leqslant 176$	$[142..146], [148..156], [158..160], 162, 164, [166..168], 170, [172..174], 176,$
$178 \leqslant s_2(q) \leqslant 206$	$178, [180..184], [186..190]_2, [192..194], [196..198], 200, 202, [204..206],$
$208 \leqslant s_2(q) \leqslant 236$	$[208..212]_2, [214..216], 218, [220..222], 224, 226, [228..230], [232..234], 236,$
$238 \leqslant s_2(q) \leqslant 298$	$238, [240..242], 244, [246..248], [250..252], [254..272]_2, [274..276], [278..298]_2,$
$300 \leqslant s_2(q) \leqslant 332$	$[300..302], [304..306], [308..316]_2, [318..320], [324..328]_2, [330..332],$
$334 \leqslant s_2(q) \leqslant 452$	$[334..404_2], [408..410], [412..416]_2, [420..432]_4, [436..440]_2, [444..452]_4,$
$454 \leqslant s_2(q) \leqslant 556$	$[454..466]_2, 472, 480, [484..488]_2, [492..498]_2, [504..528]_4, 536, 544, 548, 556,$
$564 \leqslant s_2(q) \leqslant 602$	$564, 568, 576, [580..584]_2, 588, 592, 600, 608, 644, 648, 656, 664, 680, 688, 724.$

Таблица 4

Наиболее часто встречающиеся значения $s_2(q)$

$s_2(q)$	n	m	$s_2(q)$	n	m	$s_2(q)$	n	m	$s_2(q)$	n	m
14	16310	29	40	308403	1429	72	155576	7561	124	13317	92399
16	24914	41	42	17644	2351	76	78813	10079	128	43542	55439
18	10376	97	44	191279	1259	80	182475	13441	132	12119	379^2
20	93166	71	46	13364	1681	84	39349	13859	136	32259	65519
22	49578	179	48	222011	2161	88	118659	21839	144	26015	225721
24	85598	169	50	13790	4049	92	19624	20161	152	28784	180181
26	30585	181	52	137856	3079	96	81653	22679	160	22230	151201
28	211019	239	54	19996	3361	100	36320	30241	168	13041	166319
30	23038	883	56	305278	2521	104	92798	42841	176	14319	262079
32	231756	419	60	80262	4159	108	23665	45361	200	13471	332641
34	28133	701	64	218627	5041	112	82275	78121			
36	131798	839	68	71811	5039	116	14719	81901			
38	45983	881	70	18305	20789	120	31025	87359			

В рассматриваемом диапазоне $s_2(q)$ изменяется от 5 для 3 до 724 для 61261199 и 64864799, и принимает 339 различных значений. Мы ограничились двумя таблицами, в первой из которых (табл. 3) указаны все возможные значения $s_2(q)$ (использованы обозначения $[i..j]$ для идущих подряд чисел, $[2k..2l]_2$ для идущих подряд *четных* чисел и $[4m..4n]_4$

для идущих подряд чисел, *делящихся на* 4), а во второй (табл. 4) указаны значения $s_2(q)$, которые встречаются $n > 10\,000$ раз и m — наименьшее значение q с данным $s_2(q)$..

2.2.2. Группы $PSL_2(q)$, q нечетно

По следствию 4 нужно вычислить сумму $s_3(q) = \tau((q-1)/2) + \tau((q+1)/2)$. Оказалось, что $s_3(q)$ изменяется от 3 для 3 до 602 для 64864799.

В табл. 5 будем использовать обозначения, указанные перед табл. 3.

Таблица 5

Все значения $s_3(q)$

$3 \leqslant s_3(q) \leqslant 230$	$[3..164], [166..178], [180..184], [186..210], [212..218]_2, [219..222], [224..230],$
$232 \leqslant s_3(q) \leqslant 266$	$232, 233, [234..240]_2, [241..252], [254..256], 255, 256, [258..262], [264..266],$
$268 \leqslant s_3(q) \leqslant 316$	$268, [272..290]_2, 291, [292..296]_2, 297, 298, [300..304], 306, 308, [312..316],$
$319 \leqslant s_3(q) \leqslant 360$	$319, 320, [322..324], [326..332]_2, 333, [334..344], 348, [352..356], 360,$
$362 \leqslant s_3(q) \leqslant 411$	$[362..366]_2, [368..380]_4, [382..392]_2, 393, [394..402]_2, 403, 404, 408, 409, 411,$
$412 \leqslant s_3(q) \leqslant 504$	$412, 416, 422, 424, 428, 434, 436, [440..448]_4, [450..456]_2, 482, 484, 488, 504,$
$506 \leqslant s_3(q) \leqslant 602$	$506, 508, 512, [514..516], 520, 548, 578, 580, 584, 602.$

В табл. 6 укажем значения $s_3(q)$, которые встречаются $n > 10\,000$ раз и m — наименьшее значение q с данным $s_3(q)$.

Таблица 6

Наиболее часто встречающиеся значения $s_3(q)$

$s_3(q)$	n	m	$s_3(q)$	n	m	$s_3(q)$	n	m	$s_3(q)$	n	m
8	16310	23	30	33043	43^2	54	15938	13859	88	31424	63361
10	30821	41	32	297813	1439	56	155300	12601	96	13464	122401
12	127433	79	34	63696	2399	60	39323	21121	98	19449	92399
14	57706	11^2	36	219546	2161	64	80045	13441	100	41143	55439
16	217966	13^2	38	39161	2879	66	26062	15121	104	33946	65519
18	74281	421	40	299168	2521	68	70806	21839	112	14997	100799
20	326442	239	42	20454	6047	72	71257	36721	132	11599	166319
22	63282	479	44	115823	3359	74	13455	181^2	148	10679	262079
24	333303	769	48	146398	5279	76	43984	20161			
26	66910	719	50	45882	71^2	80	54321	35281			
28	274405	31^2	52	139916	5039	84	22888	45361			

Замечание 2. Возникает обманчивое впечатление, что $s_2(q)$ и $s_3(q)$ очень просто между собой связаны, например, пропорциональны. Однако на самом деле все сложнее. Пусть

$\varepsilon = (-1)^{\frac{q-1}{2}}$. Тогда

$$q - \varepsilon \equiv 0 \pmod 4 \quad \text{и} \quad q + \varepsilon \equiv 2 \pmod 4.$$ Отсюда

$$q - \varepsilon = 2^{\alpha}\beta \text{ и } q + \varepsilon = 2\gamma, \text{ где } \alpha \geqslant 2, \text{ а } \beta \text{ и } \gamma \text{ нечетны.}$$

Таблица 7

Значения $\tau(n)$

$\tau(n)$	$o(\tau(n))$	$m(\tau(n))$	$\tau(n)$	$o(\tau(n))$	$m(\tau(n))$	$\tau(n)$	$o(\tau(n))$	$m(\tau(n))$	$\tau(n)$	$o(\tau(n))$	$m(\tau(n))$
2	2063689	2	49	4	46656	125	4	810000	245	1	29160000
3	760	4	50	3715	6480	126	1557	100800	250	12	5670000
4	6090743	6	51	7	589824	128	63439	83160	252	750	1108800
5	21	16	52	1618	61440	130	29	1658880	256	3327	1081080
6	1093224	12	54	21339	6300	132	1006	322560	260	7	11612160
7	7	128	55	6	82944	135	155	176400	264	175	3548160
8	7417249	24	56	72561	6720	136	22	6881280	270	240	1940400
9	1553	36	57	4	2359296	140	1589	181440	280	303	1995840
10	198403	48	60	99135	5040	144	61851	110880	288	4740	1441440
11	3	2^{10}	63	205	14400	147	8	1166400	294	10	8164800
12	2679516	60	64	481777	7560	150	661	226800	297	2	11289600
13	2	2^{12}	65	4	331776	152	1	27525120	300	235	2494800
14	48545	384	66	1118	46080	153	2	14745600	308	1	26127360
15	479	144	68	99	983040	154	14	3732480	312	13	14192640
16	4963136	120	70	1532	25920	156	215	1290240	315	6	6350400
17	1	2^{16}	72	247434	10080	160	22594	1166400	320	1117	2162160
18	213193	180	75	85	32400	162	1005	352800	324	244	3880800
19	1	2^{18}	76	22	3932160	165	10	2073600	330	4	14515200
20	444480	240	77	3	746496	168	7437	221760	336	610	2882880
21	195	576	78	321	184320	170	1	26542080	350	1	22680000
22	3616	3072	80	134694	15120	175	5	3240000	352	21	10644480
23	1	2^{22}	81	292	44100	176	680	967680	360	558	3603600
24	2511201	360	84	20053	20160	180	5406	277200	378	24	7761600
25	22	1296	85	1	5308416	182	3	14929920	384	757	4324320
26	1045	12288	88	3465	107520	189	42	705600	392	5	17962560
27	1083	900	90	7272	25200	192	30885	332640	396	7	17740800
28	100471	960	91	1	2985984	195	3	8294400	400	72	6486480
30	61943	720	95	1	21233664	196	94	1632960	405	3	21344400
32	1996343	840	96	284560	27720	198	86	1612800	420	26	9979200
33	49	9216	98	138	233280	200	1565	498960	432	186	7207200
34	96	196608	99	34	230400	204	2	20643840	440	1	31933440
35	18	5184	100	4916	45360	208	94	3870720	448	47	86486400
36	395896	1260	102	25	2949120	210	223	907200	450	5	17463600
38	30	786432	104	727	430080	216	6094	554400	480	97	10810800
39	26	36864	105	57	129600	220	52	2903040	486	1	31046400
40	360258	1680	108	23187	50400	224	2671	665280	504	17	14414400
42	14591	2880	110	105	414720	225	24	1587600	512	17	17297280
44	6315	15350	112	22538	60480	231	1	18662400	540	2	25225200
45	569	3600	114	5	11796480	234	17	6451200	560	1	25945920
46	3	12582912	117	14	921600	240	7070	720720	576	10	21621600
48	1174467	2520	120	47554	55440	243	17	2822400	600	1	32432400

Поэтому

$$s_2(q) = \tau(q-1) + \tau(q+1) = (\alpha+1)\tau(\beta) + 2\tau(\beta),$$
$$s_3(q) = \tau((q-1)/2) + \tau((q+1)/2) = \alpha\tau(\beta) + \tau(\beta).$$

2.3. Циклические группы и метациклические группы Фробениуса

По следствию 5 для циклических групп нужно подсчитать число делителей натуральных чисел, что было проделано в диапазоне от 2 до $2^{25} = 33\,554\,432$. Отметим, что число делителей изменяется от 2 (это в точности простые числа) до 600 для 32 432 400. В табл. 7 указаны возможные $\tau(n)$, количество $o(\tau(n))$ таких $\tau(n)$ в рассматриваемом диапазоне для n и минимальное значение $n = m(\tau(n))$ с данным $\tau(n)$.

Для метациклических групп Фробениуса можно из вычислений $\tau(n)$ можно получить по следствию 6 нужные значения $\tau(m) + \tau(n)$, где n делит $m - 1$. Однако тут возникает слишком много возможностей для m и n, что приведет к огромным таблицам, которые просто невозможно разместить в рамках одной статьи.

Заключение

Результаты, приведенные в данной работе, дают удобные для вычислений формулы рангов групп центральных единиц, что позволяет анализировать поведение рангов при возрастании параметров, таких как порядок конечного поля для линейных групп или порядок группы для циклических или метациклических групп Фробениуса. Таблицы и график, приведенные во втором разделе, показывают характер изменений параметров рангов групп центральных единиц, связанных с числом делителей чисел, которые входят в формулы для рангов.

В дальнейшем на основе полученных данных планируется изучить асимптотическое поведение рангов групп центральных единиц.

Первый автор и его ученики получали существенную техподдержку от коллективов, возглавляемых Л.Б. Соколинским, что позволяет, наконец, выразить большую признательность ему за содействие.

Литература

1. Алеев, Р.Ж. Теория центральных единиц целочисленных групповых колец групп $PSL_2(2^n)$ / Р.Ж. Алеев // Сб. научн. трудов «Комбинат. и вычислит. методы в матем.». — Омск: ОмГУ , 1999. — С. 1–19.
2. Алеев, Р.Ж. Центральные единицы целочисленных групповых колец конечных групп: дисс. . . . д-ра физ.-мат. наук. / Р.Ж. Алеев — Челябинск, 2000. — 355 с.
3. Алеев, Р.Ж. Теория групп центральных единиц целочисленных групповых колец групп $Sz(q)$ / Р.Ж. Алеев, Н.Б. Ишечкина // Труды Института математики и механики УрО РАН. — 2001. — Т.7, № 2. — С. 3–16.
4. Алеев, Р.Ж. Теорема разложения и ранги групп центральных единиц целочисленных групповых колец групп $PGL(2, q)$, q нечетно / Р.Ж. Алеев, О.В. Митина // Сибирские Электронные Математические Известия. — 2008. — Т. 5. — С. 652–672.

5. Алеев, Р.Ж. Ранги групп центральных единиц целочисленных групповых колец групп $PSL(2, q)$, q нечетно / Р.Ж. Алеев, О.В. Перавина // Вестник ЧелГУ, сер. «Математика. Механика». — 1999, № 1(4). — С. 5–15.
6. Василенко, О.Н. Теоретико–числовые алгоритмы в криптографии. / О.Н. Василенко — М.: МЦНМО, 2006. — 336 с.
7. Глухов, М.М. Введение в теоретико–числовые методы в криптографии. / М.М. Глухов, И.А. Круглов, А.Б. Пичкур, А.В. Черемушкин — СПб.: Изд-во «Лань», 2011. — 400 с.
8. Кэртис, Ч. Теория представлений конечных групп и ассоциативных алгебр: Пер. с англ. / Ч. Кэртис, И. Райнер — М.: Наука, 1969. — 668 с.
9. Шумакова, Е.О. Группы центральных единиц целочисленных групповых колец метациклических групп Фробениуса / Е.О. Шумакова // Сибирские Электронные Математические Известия. — 2008. — Т. 5. — С. 691–698.
10. Aleev, R. Ž. Higman's central unit theory, units of integral group rings of finite cyclic groups and Fibonacci numbers / R. Ž. Aleev // Intern. Journ. Algebra and Computations. — 1994. — Vol. 4. — P. 309–358.
11. Ferraz, R. A. Simple components and central units in group rings / R.A. Ferraz // Journal of Algebra. — 2004. — Vol. 279, No. 1. — P. 91–203.
12. The GAP Group, GAP - Groups, Algorithms, and Programming, Version 4.7.4. — 2014. URL: `http://www.gap-system.org` (дата обращения: 04.05.2014).
13. Ritter, J. Trivial units in RG / J. Ritter, S.K. Sehgal // Mathematical Proceedings of the Royal Irish Academy. — 2005. — Vol. 105A, No. 1. — P. 25–39.
14. Schönert, M. GAP – Groups, Algorithms, and Programming / M. Schönert et al. — Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, sixth edition, 1997. URL: `http://www.gap-system.org/Gap3` (дата обращения: 14.06.2012).

Алеев Рифхат Жалялович, д.ф.-м.н., профессор кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), aleevrz@susu.ac.ru.

Цыбина Наталья Андреевна, магистр кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), tsybinanatasha@gmail.com.

Поступила в редакцию 20 декабря 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 1, pp. 71–85

DOI: 10.14529/cmse150107

THE COMPUTATION OF RANKS OF UNIT GROUPS OF INTEGRAL GROUP RINGS OF FINITE GROUPS

R.Zh. Aleev, South Ural State University (Chelyabinsk, Russian Federation)
aleevrz@susu.ac.ru,
N.A. Tsybina, South Ural State University (Chelyabinsk, Russian Federation)
tsybinanatasha@gmail.com

The study of central units (central invertible elements) of integral group rings is encountered to difficult calculations almost everywhere, both in the case of finding of individual central unit and in the case of describing of group of central elements. By virtue of torsion part of central unit group is trivial (up to sign those are elements of center group) it is more interesting to find data about torsion free part that is direct product infinite cyclic groups. The number of such infinite factors is the rank of central unit group. Therefore the ranks of central unit groups of integral group rings of finite groups are the very important characteristic those groups. So that the computation ranks of central unit groups has big interest for study of central unit groups. In the paper we point out the formulas for computation of ranks in general case and some important particular cases. On the base of those formulas we compute the ranks in quite large ranges. We used computer algebra system GAP. The results are shown on tables and graph.

Keywords: group character, central unit, rank of Abelian group, system GAP.

References

1. Aleev R.Zh. Teoriya central'nyh edinic celochislennyh gruppovyh kolec grupp $PSL_2(2^n)$ [The theory of central units of integral goup rings of groups $PSL_2(2^n)$] // Sb. nauchn. trudov «Combinat. i vychislit. metody v matem.». Omsk: OmGU , 1999. P. 1–19.
2. Aleev R.Zh. Central'nye edinicy celochislennyh gruppovyh kolec konechnyh grupp [Central units of integral group rings of finite groups]: diss. . . . d-ra fis.-mat. nauk. Chelyabinsk, 2000. 355 p.
3. Aleev R.Zh., Ishechkina N.B. A Theory of Central Unit Groups of Integral Group Rings of Groups $Sz(q)$ // Proccedings of the Steklov Institute of Mathematics, Suppl. 2. MAIK «Nauka/Interperiodica». 2001. P. 1–15.
4. Aleev R. Zh., Mitina O. V. Teorema razlozheniya and rangi central'nyh edinic celochislennyh gruppovyh kolec grupp $PSL(2, q)$, q nechetno [The decomposition theorem and ranks of central unit groups of integer group rings of groups $PGL2(q)$, q odd] // Siberian Electonic Mathematcal Reports. 2008. Vol. 5. P. 652–672
5. Aleev R.Zh., Peravina O.V. Rangi grupp central'nyh edinic celochislennyh gruppovyh kolec grupp $PSL(2, q)$, q nechetno [The ranks central units of integral group rings of finite groups $PSL(2, q)$, q odd] // Vest. ChelSU. ser. Mat. Mekh., 1999, No 1(4). P. 5–15.
6. Vasilenko O.N. Number-theoretic Algorithms in Cryptography. AMS, 2007. 243 p.

7. Glukhov M.M., Kruglov I.A., Pichkur A.B., Cheremushkin A.V. Vvedenie v teoretiko-chislovye metody kriptografii [Introduction to theoretical and numerical methods in cryptography]. Sankt-Peterburg: Lan', 2011. 400 p.
8. Curtis, Charles W.; Reiner, Irving, Representation theory of finite groups and associative algebras, Pure and Applied Mathematics, Vol. XI, New York-London, Interscience Publishers, a division of John Wiley & Sons, 1962. 703 p.
9. Shumakova E.O. Gruppy central'nyh edinic celochislennyh gruppovyh kolec metaciklicheskih grupp Frobeniusa [Central units in integral group rings for Frobenius metacyclic groups] // Siberian Electonic Mathematcal Reports. 2008. Vol. 5. C. 691–698.
10. Aleev R. Ž. Higman's central unit theory, units of integral group rings of finite cyclic groups and Fibonacci numbers // Intern. Journ. Algebra and Computations. 1994. Vol. 4. P. 309–358.
11. Ferraz R. A. Simple components and central units in group rings // Journal of Algebra. 2004. Vol. 279, No. 1. P. 91–203.
12. The GAP Group, GAP - Groups, Algorithms, and Programming, Version 4.7.4. — 2014. URL: `http://www.gap-system.org` (accessed: 04.05.2014).
13. Ritter J., Sehgal S.K. Trivial units in RG // Mathematical Proceedings of the Royal Irish Academy. 2005. Vol. 105A, No 1. P. 25–39.
14. Schönert M. et al. GAP – Groups, Algorithms, and Programming // Lehrstuhl D für Mathematik, Rheinisch Westfälische Technische Hochschule, Aachen, Germany, sixth edition, 1997. URL: `http://www.gap-system.org/Gap3` (accessed: 14.06.2012).

Received December 20, 2014.

УДК 517.948 DOI: 10.14529/cmse150108

КОНЕЧНОРАЗНОСТНАЯ АППРОКСИМАЦИЯ МЕТОДА РЕГУЛЯРИЗАЦИИ А.Н. ТИХОНОВА n-го ПОРЯДКА

В.П. Танана, С.И. Белъков

Статья является естественным продолжением работы А.Н. Тихонова, в которой впервые была сформулирована идея конечномерного приближения регуляризующей задачи, однако условия, накладываемые на операторы являются трудно проверяемыми. В настоящей работе предложено другое условие, которое легче использовать на практике и с его помощью произведено доказательство теоремы о сходимости конечноразностных аппроксимаций метода регуляризации Тихонова к точному решению регуляризованной задачи. Применение предложенного метода конечноразностных приближений продемонстрировано на примере интегрального уравнения Фредгольма первого рода.

Ключевые слова: обратная задача, регуляризация, конечно-разностная аппроксимация, некорректная задача, интегральное уравнение.

Введение

Многие задачи, имеющие прикладное значение (такие как геологоразведка или определение температур внутри ракетных двигателей), являются некорректно поставленными и требуют использования аппарата регуляризации для их решения. Настоящая статья является естественным продолжением работы [1]. Дело в том, что в [1] для доказательства сходимости конечномерных аппроксимация использовано условие слабой замкнутости пары $A,\{A_k\}$, где A — оператор исходной задачи, а A_k — операторы аппроксимирующих задач. Это условие, впервые предложенное в [2] и являющееся наиболее общим при исследовании вопросов сходимости конечномерных аппроксимаций, трудно проверяемо. Поэтому в настоящей работе предложено другое условие и с его помощью доказана теорема о сходимости конечноразностных аппроксимация метода регуляризации Тихонова n-го порядка, которая была сформулирована в работе [3], но не доказана.

Статья состоит из трех частей. В первой части вводятся ограничения на рассматриваемые объекты, приводится формулировка решаемой задачи и ее регуляризация сведением к вариационной задаче. Во второй части работы производится построение последовательных конечномерных приближений регуляризованной задачи и доказательство соответствующих теорем сходимости. Замена бесконечномерных регуляриязующих вариационных задач конечномерными аналогами позволяет построить последовательность численных приближенных решений, которая сходится к регуляризованному решению. В третьей части рассматривается пример построения конечноразностной аппроксимации метода регуляризации А.Н. Тихонова n-го порядка для оператора Фредгольма.

1. Постановка задачи и метод регуляризации

Пусть H — сепарабельное гильбертово пространство, A — линейный замкнутый оператор с областью определения $D(A)$, всюду плотной в H и областью значения $R(A)$ из H, T — линейный замкнутый оператор с областью определения $D(T)$, всюду плотной в H и множеством значений $R(T)$ из H.

Предположим, что существует число $c > 0$ такое, что для любого $u \in D(T)$ имеет место соотношение

$$\|Tu\| \geq c\|u\|. \tag{1}$$

Лемма 1. Пусть n — некоторое натуральное число, большее единицы. Тогда оператор T^n будет замкнут.

Доказательство. Предположим, что $\{u_k\} \subset D(T)$

$$u_k \to \hat{u} \tag{2}$$

и

$$T^n u_k \to \overline{f}_n. \tag{3}$$

Тогда из условий (1), (3) следует, что $T^{n-1}u_k \to \overline{f}_{n-1}$.

Продолжая рассуждения аналогичным образом, приходим к тому, что

$$Tu_k \to \overline{f}_1 \tag{4}$$

Так как оператор T замкнут, то из соотношений (2) и (4) следует, что $\hat{u} \in D(T)$ и $\overline{f}_1 = T\hat{u}$. Продолжая этот процесс, окончательно получим, что $\hat{u} \in D(T^n)$ и $\overline{f}_n = T^n\hat{u}$. Тем самым лемма доказана.

Рассмотрим операторное уравнение

$$Au = f, \quad u \in D(A), \quad f \in H. \tag{5}$$

Предположим, что при $f = f_0$ существует точное решение u_0 уравнения (5) и $u_0 \in D(T^n)$, но вместо f_0 нам известны $f_\delta \in H$ и уровень погрешности $\delta > 0$ такие, что $\|f_\delta - f_0\| \leq \delta$.

Требуется по исходной информации (f_δ, δ) найти приближенное решение u_δ, сходящееся к точному. Метод регуляризации (см. [3]) поставленной задачи заключается в сведении ее к вариационной

$$\inf\left\{\|Au - f_\delta\|^2 + \alpha\|T^n u\|^2 : \quad u \in D(A) \cap D(T^n)\right\}, \tag{6}$$

где $\alpha > 0$.

Вариационная задача (6) разрешима единственным образом (см. [4], стр. 73). Обозначим решение вариационной задачи (6) через u_δ^α и будем называть его приближенным решением уравнения (1), полученным методом регуляризации, или регуляризованным решением.

2. Конечномерная аппроксимация метода регуляризации

Пусть $\{H_k\}$ — возрастающая последовательность конечномерных подпространств пространства H, A_k и T_k — линейные ограниченные операторы, отображающие пространство H_k в H и H_k соответственно.

Конечномерная аппроксимация метода регуляризации заключается в замене вариационной задачи (6) ее конечномерным аналогом

$$\inf\left\{\|A_k u - f_\delta^k\|^2 + \alpha\|T_k^n u\|^2 : \quad u \in H_k\right\}. \tag{7}$$

Аппроксимирующая задача (7) так же, как и (6), разрешима единственным образом. Решение этой задачи в дальнейшем будем обозначать через $u_\delta^\alpha(k)$. Главным вопросом, возникающим при этом, является вопрос о сходимости конечномерных аппроксимаций $u_\delta^\alpha(k)$ к регуляризованному решению u_δ^α при $k \to \infty$.

Для ответа на поставленный вопрос введем ряд вспомогательных определений.

Пусть B, S, B_k, S_k — линейные замкнутые операторы с областями определения $D(B), D(S), D(B_k), D(S_k)$, всюду плотными в H и областями значений из H.

Определение 1. Последовательности операторов $\{B_k\}$ и $\{S_k\}$ будем называть B и S полными одновременно, если $\overline{D(B) \cap D(S)} = H$ и для любого $u \in D(B) \cap D(S)$ найдется последовательность $\{u_k\}$, $u_k \in D(B_k) \cap D(S_k)$ такая, что $u_k \to u$, $B_k u_k \to Bu$ и $S_k u_k \to Su$ (см. [4], стр. 113).

Определение 2. Будем говорить, что пара $\{B_k\}, \{S_k\}$ удовлетворяет условию дополнительности, если существует $\gamma \neq 0$ такое, что $\forall u_k \in D(B_k) \cap D(S_k) \quad \|B_k u_k\|^2 + \|S_k u_k\|^2 \geq \gamma^2 \|u_k\|^2$.

Определение 3. Пару $B, \{B_k\}$ будем называть слабо замкнутой, если из того, что $u_k \xrightarrow{w} \hat{u}$, а $B_k u_k \xrightarrow{w} \overline{f}$ следует, что $\hat{u} \in D(B)$ и $\overline{f} = B\hat{u}$ (см. [4], стр. 113).

Теорема 1. Пусть $f_\delta^k \to f_\delta$, а пара $\{A_k\}, \{T_k^n\}$ удовлетворяет условию дополнительности. Тогда, если последовательности $\{A_k\}$ и $\{T_k^n\}$ являются A и T^n полными одновременно, а последовательности $\{A_k'\}$ и $\{(T_k^n)'\}$ — A' и $(T^n)'$ полными одновременно, то имеет место сходимость конечномерных аппроксимаций $u_\delta^\alpha(k)$ к регуляризованному решению u_δ^α при $k \to \infty$.

Доказательство. См. [4], стр. 114.

Обозначим через B' оператор, сопряженный с B, а через B_k' оператор, сопряженный с B_k, через G обозначим множество такое, что $G \subset D(B')$ и всюду плотно в $D(B')$ относительно B'-нормы, где

$$\|g\|_{B'}^2 = \|g\|_H^2 + \|B'g\|_H^2 \tag{8}$$

Лемма 2. Если для любого k имеет место $G \subset D(B_k')$ и для любого $g \in G$ выполнено $B_k' g \to B'g$, то пара $B, \{B_k\}$ слабо замкнута.

Доказательство. Пусть дана последовательность $\{u_k\} \subset H$ такая, что для любого k имеет место $u_k \in D(B_k)$ и

$$u_k \xrightarrow{w} \hat{u} \tag{9}$$

и

$$B_k u_k \xrightarrow{w} \overline{f} \tag{10}$$

Тогда для любого $g \in G$ справедливо соотношение

$$(g, B_k u_k) \to (g, \overline{f}) \tag{11}$$

или

$$\left(B_k' g, u_k\right) \to \left(B' g, \hat{u}\right) = \left(g, B\hat{u}\right) \tag{12}$$

Из соотношений (11) и (12) следует, что для любого $g \in G$

$$\left(g, \overline{f}\right) = \left(g, B\hat{u}\right) \tag{13}$$

Так как множество G всюду плотно в H, то из (13) следует, что $\hat{u} \in D\left(B\right)$ и $B\hat{u} = \overline{f}$.

Тем самым лемма доказана.

Лемма 3. Пусть для любого k область значений $R\left(B_k\right)$ оператора B_k содержится в его области определения $D\left(B_k\right)$. Тогда, если пара $B, \left\{B_k\right\}$ слабо замкнута и существует $\gamma \neq 0$ такое, что для любых k и $u \in D\left(B_k\right)$ выполняется $\left\|B_k u\right\|^2 \geq \gamma^2 \left\|u\right\|^2$, то для любого натурального n пара $B^n, \left\{B_k^n\right\}$ слабо замкнута.

Доказательство. Будем доказывать лемму по индукции.

При $n = 1$ справедливость леммы следует из ее предположений.

Допустим, что лемма справедлива для всех n от 1 до $m-1$ и докажем справедливость ее утверждений при $n = m$.

Пусть дана последовательность $\left\{u_k\right\}$ такая, что для любого k $u_k \in D\left(B_k^m\right)$,

$$u_k \xrightarrow{w} \hat{u} \tag{14}$$

и

$$B_k^m u_k \xrightarrow{w} \overline{f}. \tag{15}$$

Тогда из условий леммы и соотношения (15) следует ограниченность последовательности $\left\{B_k^{m-1} u_k\right\}$, а, следовательно, и ее слабая компактность, то есть без ограничения общности можем считать, что

$$B_{k_l}^{m-1} u_{k_l} \xrightarrow{w} \hat{\varphi}. \tag{16}$$

Из предположения индукции, (14) и (16) следует, что $\hat{u} \in D\left(B^{m-1}\right)$ и

$$B^{m-1}\hat{u} = \hat{\varphi}, \tag{17}$$

а из предположений леммы, (15) и (16) следует, что $\hat{\varphi} \in D\left(B\right)$ и

$$\overline{f} = B\hat{\varphi}. \tag{18}$$

Из (17) и (18) следует утверждение леммы, то есть, что $\hat{u} \in D\left(B^m\right)$ и $\overline{f} = B^m \hat{u}$.

Тем самым лемма доказана.

Пусть $G \subset D\left(A'\right) \cap D\left(T'\right)$ и при этом всюду плотно в $D\left(A'\right)$ и $D\left(T'\right)$ относительно A'-нормы и T'-нормы соответственно. Тогда справедлива теорема.

Теорема 2. Пусть $f_\delta^k \to f_\delta$, а последовательности $\left\{A_k\right\}$ и $\left\{T_k^n\right\}$ являются A и T^n полными одновременно. Тогда если для любого $g \in G$ выполнено $A_k' g \to A' g$ и $T_k' g \to T' g$, то имеет место сходимость конечноразностных аппроксимаций $u_\delta^\alpha\left(k\right)$ к регуляризованному решению u_δ^α при $k \to \infty$.

Доказательство. Эта теорема следует из теоремы 1 и лемм 1, 2, 3.

3. Конечноразностная аппроксимация метода регуляризации Тихонова n-го порядка при решении интегрального уравнения

Пусть $H = L_2[0,1]$, а

$$Au = \int_0^1 K(s,t)\,u(s)\,ds, \quad 0 \le t \le 1, \tag{19}$$

где $K \in C[0,1;0,1]$, $u \in L_2[0,1]$, $Au \in L_2[0,1]$.

Рассмотрим разбиение отрезка $[0,1]$ на 2^k равных частей точками $0 = s_0 < s_1 < s_2 < \ldots < s_{2^k} = 1$, где для любого i, $i = 1,2,\ldots,2^k$ $s_i - s_{i-1} = \dfrac{1}{2^k}$.

Обозначим через Δ_i полуинтервал деления $\Delta_i = [s_{i-1}, s_i)$, а через U_k подпространство $L_2[0,1]$, состоящее из кусочно-постоянных функций, то есть

$$U_k = \{u(s) : u(s) = u_i,\ s \in \Delta_i,\ i = 1,2,\ldots,2^k\}. \tag{20}$$

Пусть P_k оператор метрического проектирования пространства $L_2[0,1]$ на U_k. Оператор P_k самосопряжен, то есть

$$P_k' = P_k, \tag{21}$$

где P_k' — оператор, сопряженный к P_k, и для любого $u \in L_2[0,1]$

$$P_k u \to u \text{ при } k \to \infty. \tag{22}$$

Следуя [4], стр. 119

$$A_k u = \int_0^1 \overline{K_k}(s,t)\,u(s)\,ds = \frac{1}{2^k}\sum_{i=1}^{2^k} K_{ij} u_i, \tag{23}$$

где $u \in U_k$, определенному (20), а $\overline{K_k}(s,t) = K\left(\dfrac{i}{2^k}; \dfrac{j}{2^k}\right)$ при $s \in \Delta_i$ и $t \in \Delta_j$; $K_{ij} = K\left(\dfrac{i}{2^k}; \dfrac{j}{2^k}\right)$.

Если рассмотреть расширение $\overline{A_k}$ оператора A_k с U_k на все пространство $L_2[0,1]$, то следуя [4]

$$\left\|\overline{A_k} - A\right\| \to 0 \text{ при } k \to \infty. \tag{24}$$

Пусть оператор T определен формулой

$$Tu(t) = \frac{d}{dt}\left[u(t)\right], \tag{25}$$

где $D(T) = \{u(t) : u(t),\ u'(t) \in L_2[0,1],\ u(0) = 0\}$.

Оператор T — линейный замкнутый с областью определения $D(T)$, всюду плотной в $L_2[0,1]$.

Оператор T', сопряженный T, будет иметь вид

$$T'g(t) = -\frac{d}{dt}\left[g(t)\right], \tag{26}$$

где $D(T') = \left\{g(t) : g(t),\, g'(t) \in L_2[0,1],\, g(1) = 0\right\}$.

Кроме того, легко проверить, что для любого $u \in D(T)$ оператор T удовлетворяет соотношению

$$\|Tu\| \geq \|u\|. \tag{27}$$

На пространстве U_k, определенном формулой (20), определим оператор T_k следующим образом

$$T_k u(t) = \begin{cases} 0, & t \in \Delta_1 \\ \dfrac{u(t) - u\left(t - 2^{-k}\right)}{2^k}, & t \notin \Delta_1 \end{cases}, \tag{28}$$

где $D(T_k) = \left\{u(t) : u(t) \in U_k,\, u(t) = 0,\, t \in [0,1]\right\}$.

Из (28) следует, что $D(T_k)$ является подпространством U_k, следовательно, и $L_2[0,1]$, кроме того оператор T_k непрерывен на $D(T_k)$ и область значений $R(T_k)$ оператора T_k содержится в $D(T_k)$, а из соотношения (28) легко следует, что для любого $u \in D(T_k)$

$$\|T_k u\| \geq \|u\|. \tag{29}$$

Для любого k сопряженный оператор T_k' будет иметь вид

$$T_k' = \overline{T_k} P_k, \tag{30}$$

где P_k оператор метрического проектирования $L_2[0,1]$ на U_k, а $\overline{T_k}$ определен формулой

$$\overline{T_k} g(t) = \begin{cases} 0, & t \in \Delta_{2^k} \\ \dfrac{g(t) - g\left(t + 2^{-k}\right)}{2^k}, & t \notin \Delta_{2^k} \end{cases}, \tag{31}$$

где $g \in U_k$.

В качестве множества G, использованного в лемме 2, возьмем следующее множество:

$$\begin{aligned} G = \Big\{ & g(t) : g(t) \in C[0,1],\, g(t) = a_i t + b_i,\, t \in \Delta_i, \\ & g(t) = 0,\, t \in \Delta_1 \cup \Delta_{2^k},\, i = 1,2,\ldots,2^k,\, k = 1,2,\ldots \Big\} \end{aligned}, \tag{32}$$

Тогда множество G содержится в области определения $D(T')$ оператора T'.

Определение 4. Определим срезающую функцию $\xi(t)$:

$$\xi(t)=\begin{cases}1, & t\in\left[\frac{7}{4}\delta;1-\frac{7}{4}\delta\right],\\ \frac{2}{\delta}\left(t-\frac{5}{4}\delta\right), & t\in\left[\frac{5}{4}\delta;\frac{7}{4}\delta\right],\\ -\frac{2}{\delta}\left(t-1+\frac{5}{4}\delta\right), & t\in\left[1-\frac{7}{4}\delta;1-\frac{5}{4}\delta\right],\\ 0, & t\in[0;1]\setminus\left[\frac{5}{4}\delta;1-\frac{5}{4}\delta\right].\end{cases}$$

Ее обобщенная производная обладает свойством (см. [5], стр. 119):

$$\left|\xi_h'(t)\right|=\left|(\xi_h)'(t)\right|=\left|\int\limits_{|t-v|\le h}\xi'(v)\,dv\right|\le\frac{2}{\delta}\quad\forall t\in[0;1],\quad\forall h\in\left[0;\frac{\delta}{4}\right],\tag{33}$$

Лемма 4. Множество G содержится в области определения $D(T')$ оператора T', определенного (26) и всюду плотно в $D(T')$ относительно T'-нормы.

Доказательство. Из (26) и (32) следует, что множество $G\subset D(T')$. Докажем, что оно всюду плотно в нем относительно T'-нормы.

Рассмотрим множество $D=\left\{h(t):h(t)\in C^\infty[0;1],\,h(0)=h(1)=0\right\}$.

Будем доказывать, что D всюду плотно в $D(T')$.

Возьмем произвольную функцию $h(t)\in H_0^1(0;1)$ и срезающую функцию $\xi_\delta(t)$ со свойством

$$\left|\xi_\delta'(t)\right|\le\frac{c}{\varepsilon}\quad\forall t\in[0;1].$$

Согласно теореме о дифференцировании обобщенной производной (см. [5], стр. 118) функция $h_\delta(t)=h(t)\xi_\delta(t)$ имеет обобщенную производную $h_\delta'(t)=h'(t)\xi_\delta(t)+h(t)\xi_\delta'(t)$. Кроме того, ясно, что $h_\delta(t)$ финитна на $[0;1]$ и $h_\delta(t)\in H_0^1(0;1)$. Убедимся, что $\left\|h_\delta(t)-h(t)\right\|_{H^1(0;1)}\to0$ при $\delta\to0$. Зададим произвольное число $\varepsilon>0$. Тогда $\exists\delta_0>0:\ \forall\delta,\,0<\delta<\delta_0$,

$$\left\|h_\delta(t)-h(t)\right\|_{L^2(0;1)}=\int\limits_0^1|h(t)|^2\left(1-\xi_\delta(t)\right)^2dt\le$$

$$\le\int\limits_0^{2\delta}|h(t)|^2\,dt+\int\limits_{1-2\delta}^1|h(t)|^2\,dt<\varepsilon^2$$

в силу абсолютной непрерывности интеграла Лебега. Далее, из формулы Ньютона-Лейбница с учетом условия $h(0)=h(1)=0$ имеем

$$h(t)=\int\limits_0^t h'(\xi)\,d\xi=\int\limits_{1-t}^1h'(\xi)\,d\xi\,.$$

Отсюда помощью неравенства Коши–Буняковского получим:

$$|h(t)|^2\le t\int\limits_0^t|h'(\xi)|^2\,d\xi,$$

$$|h(t)|^2\le(1-t)\int\limits_t^1|h'(\xi)|^2\,d\xi\quad\forall t\in[0;1].$$

С учетом свойств выбранной срезающей функции, формулы для $h_\delta{}'(t)$, абсолютной непрерывности интеграла Лебега и последних неравенств будем иметь

$$\|h_\delta(t)-h(t)\|_{L^2(0;1)}=\left(\int_0^1\left|h'(t)-h'(t)\xi_\delta(t)-h(t)\xi_\delta{}'(t)\right|^2dt\right)^{\frac12}\le$$

$$\le\left(\int_0^1|h'(t)|^2\left(1-\xi_\delta(t)\right)^2dt\right)^{\frac12}+\left(\int_0^1|h(t)|^2\left|\xi_\delta{}'(t)\right|^2dt\right)^{\frac12}=$$

$$=\left(\int_0^{2\delta}|h'(t)|^2\left(1-\xi_\delta(t)\right)^2dt+\int_{1-2\delta}^1|h'(t)|^2\left(1-\xi_\delta(t)\right)^2dt\right)^{\frac12}+$$

$$+\left(\int_\delta^{2\delta}t\left(\int_0^t|h'(\xi)|^2d\xi\right)\frac{c}{\delta^2}{}^2dt+\int_{1-2\delta}^{1-\delta}(1-t)\left(\int_t^1|h'(\xi)|^2d\xi\right)\frac{c}{\delta^2}{}^2dt\right)^{\frac12}\le$$

$$\le\varepsilon+c\left(2\int_0^{2\delta}|h'(\xi)|^2d\xi+2\int_{1-2\delta}^1|h'(\xi)|^2d\xi\right)^{\frac12}<2\varepsilon$$

$\forall\delta,\ 0<\delta<\delta_0$.

Таким образом, получаем оценку $\|h_\delta(t)-h(t)\|_{H^1(0;1)}<3\varepsilon\quad\forall\delta,\ 0<\delta<\delta_0$.

Зафиксируем одно из таких значений δ и возьмем среднюю функцию $(h_\delta)_h(t)$. Тогда при $h\to0$ в силу свойств средних функций (см. [5], стр. 113, 117) получим

$$\|(h_\delta)_h(t)-h_\delta(t)\|_{L^2(0;1)}\to0,\quad\left\|(h_\delta)'_h(t)-h'_\delta(t)\right\|_{L^2(0;1)}=\left\|(h_\delta)'_h(t)-h'_\delta(t)\right\|_{L^2\left(\frac{\delta}{2};1-\frac{\delta}{2}\right)}\to0,$$

поэтому

$$\|(h_\delta)_h(t)-h_\delta(t)\|_{H^1(0;1)}<\varepsilon\quad\forall h,\quad0<h<h_0<\delta.$$

Таким образом, $\|h_\delta(t)-h(t)\|_{H^1(0;1)}\to0$.

Т.к. G всюду плотно в D, т.е. $\overline{G}=D$, а D всюду плотно в $D(T')$, т.е. $\overline{D}=D(T')$, то $\overline{G}=D(T')$ и тем самым лемма доказана.

Лемма 5. Пусть пара $T',\{T'_k\}$ определена формулами (26), (30) и (31), а множество G — формулой (32). Тогда для любого $g\in G$ имеет место соотношение

$$T'_kg\to T'g.$$

Доказательство. Пусть $g_0\in G$. Тогда найдется номер k_0 такой, что

$$g_0(t)=a_i^0t+b_i^0,\ t\in\Delta_i,\ i=1,2,\dots,2^{k_0}. \tag{34}$$

Из (34) следует, что

$$T'g_0(t)=-a_i^0,\ t\in\Delta_i,\ i=1,2,\dots,2^{k_0}. \tag{35}$$

Теперь возьмем произвольный интервал $\Delta_{i_0}=\left[t_{i_0-1},t_{i_0}\right)$ и разобьем его на 2^m равных частей. Обозначим точки деления при этом следующим образом

$$t_{i_0-1}=\overline{t_0}<\overline{t_1}<\overline{t_2}<\dots<\overline{t_{2^m-1}}<\overline{t_{2^m}}=t_{i_0},$$

а полуинтервалы через Δ_{i_0j}, где $\Delta_{i_0j}=\left[\overline{t_{j-1}},\overline{t_j}\right)$, $j=1,2,\dots,2^m$ и $\overline{t_j}-\overline{t_{j-1}}=2^{-(k_0+m)}$.

Из (34) следует, что

$$P_{k_0+m}g_0(t) = \left\{ a_{i_0}^0 \cdot \frac{\overline{t_j} + \overline{t_{j-1}}}{2} + b_{i_0}^0 : t \in \Delta_{i_0 j} \right\}. \tag{36}$$

Из (30), (31) и (36) следует, что

$$T'_{k_0+m}g_0(t) = -a_{i_0}^0,\ t \in \Delta_i \setminus \left(a_{i_0 1} \cup a_{i_0 2^m} \right). \tag{37}$$

Пусть $\overline{a_0} = \max\limits_i \left\{ \left| a_i^0 \right| : i = 1,2,\ldots,2^{k_0} \right\}$, тогда

$$\left\| T'g_0 - T'_{k_0+m}g_0 \right\| \le \frac{\overline{a_0}^2}{2^{m-5}}. \tag{38}$$

Из соотношений (38) следует, что $T'_{k_0+m}g_0 \to T'g_0$ при $m \to \infty$.

Лемма 6. Пусть пара $T, \{T_k\}$ определена формулами (25) и (28), а n — произвольное натуральное число. Тогда последовательность операторов $\left\{ T_k^{\,n} \right\}$ является T^n-полной.

Доказательство. Будем доказывать лемму по индукции.

Пусть $n = 1$ и u_0 — произвольный элемент из $D(T)$. Тогда, обозначив элемент Tu_0 через v_0^1, построим последовательность $\left\{ v_k^1 \right\}$ следующим образом:

$$v_k^1 = P_k v_0^1,$$

где P_k — оператор метрического проектирования пространства $L_2[0,1]$ на U_k, $v_k^1 \in U_k$, а подпространство U_k определено формулой (20).

Из (22) следует, что $v_k^1 \to v_0^1$.

Тогда по произвольно взятому $\varepsilon > 0$ можно указать номер k_0 такой, что $\left\| v_{k_0}^1 - v_0^1 \right\| < \frac{\varepsilon}{4}$.

Теперь выберем число m_0 такое, что функция $\overline{v}_{k_0}^1(t)$, определенная следующим образом

$$\overline{v}_{k_0}^1(t) = \begin{cases} 0, & 0 \le t < \dfrac{1}{2^{k_0+m_0}} \\ v_{k_0}^1(t), & t \ge \dfrac{1}{2^{k_0+m_0}} \end{cases}. \tag{39}$$

будет удовлетворять соотношению

$$\left\| \overline{v}_{k_0}^1 - v_0^1 \right\| < \frac{\varepsilon}{2}. \tag{40}$$

Из определения $\overline{v}_{k_0}^1$ следует $\overline{v}_{k_0}^1 \in U_{k_0+m_0}$.

Обозначим через $\overline{u}_{k_0}^1$ элемент, равный $T^{-1}\overline{v}_{k_0}^1$, который будет непрерывен и кусочно линеен, то есть

$$\overline{u}_{k_0}^1 = \left\{ a_i t + b_i : t \in \Delta_i,\ i = 1,2,\ldots,2^{k_0+m_0} \right\}. \tag{41}$$

Из (27) и (40) будет следовать, что

$$\left\| \overline{u}_{k_0}^1 - u_0 \right\| < \frac{\varepsilon}{2}. \tag{42}$$

Теперь каждый из полуинтервалов $\Delta_i = \left[t_{i-1}, t_i\right)$ разобьем на 2^p равных частей, где p — некоторое натуральное число. При этом точки деления обозначим следующим образом:

$$t_{i-1} = \overline{t_0} < \overline{t_1} < \overline{t_2} < \ldots < \overline{t_{2^p}} = t_i\,,$$

а соответствующие интервалы через Δ_{ij}, где $\Delta_{ij} = \Delta_i = \left[\overline{t_{j-1}}, \overline{t_j}\right)$, $j = 1, 2, \ldots, 2^p$ и $\overline{t_j} - \overline{t_{j-1}} = \dfrac{1}{2^{k_0+m_0+p}}$.

По функции $\overline{u}_{k_0}(t)$, определенной формулой (41), построим функцию $\overline{u}_{k_0+m_0+p}(t)$ следующим образом

$$\overline{u}_{k_0+m_0+p}(t) = \left\{\overline{u}_{k_0+m_0+p}(t) = \overline{u}^1_{k_0}\left(\overline{t_j}\right) : t \in \Delta_{ij},\ i = 1, 2, \ldots, 2^{k_0+m_0},\ j = 1, 2, \ldots, 2^p\right\}. \tag{43}$$

Из определения оператора T_k (см. (28)) и (39), (43) следует, что $\overline{u}_{k_0+m_0+p}(t) \in U_{k_0+m_0+p} \cap D\left(T_{k_0+m_0+p}\right)$ и

$$T_{k_0+m_0+p}\overline{u}_{k_0+m_0+p} = \overline{v}^1_{k_0}\,. \tag{44}$$

Так как функция $\overline{u}^1_{k_0}(t)$, определенная формулой (41), непрерывна, то из (43) следует, что

$$\overline{u}_{k_0+m_0+p} \to \overline{u}^1_{k_0} \text{ при } p \to \infty\,.$$

Таким образом, найдется номер p_0 такой, что для любого $p \geq p_0$

$$\left\|\overline{u}_{k_0+m_0+p} - \overline{u}^1_{k_0}\right\| < \frac{\varepsilon}{2}\,.$$

Тем самым для $n = 1$ лемма доказана.

Предположим, что лемма справедлива для $n \leq l-1$ такой, что и докажем ее для $n = l$.

Из предположения индукции следует, что для произвольного элемента $u_0 \in D\left(T^l\right)$ найдется последовательность $\{v_k\}$ такая, что для любого k $v_k \in D\left(T_k^{l-1}\right)$ и

$$v_k \to Tu_0 \tag{45}$$

и

$$T_k^{l-1}v_k \to T^l u_0 \tag{46}$$

Из (45) и (46) следует, что по произвольно взятому $\varepsilon > 0$ найдется номер $\overline{k}_0$ такой, что при $k \geq \overline{k}_0$

$$\left\|v_{\overline{k}_0} - Tu_0\right\| < \frac{\varepsilon}{2} \tag{47}$$

и

$$\left\|T_k^{l-1}v_k - T^l u_0\right\| < \frac{\varepsilon}{2}\,. \tag{48}$$

Обозначим через $u_{\overline{k}_0}$ функцию, равную $T^{-1}v_{\overline{k}_0}$, которая будет непрерывной и кусочно-линейной, то есть

$$u_{\overline{k}_0}(t) = \left\{\overline{a}_i t + \overline{b}_i : t \in \Delta_i,\ i = 1, 2, \ldots, 2^{\overline{k}_0}\right\}. \tag{49}$$

Из (27) и (47) будет следовать, что

$$\hat{C}\left\|u_{\bar{k}_0} - u_0\right\| < \frac{\varepsilon}{2}. \tag{50}$$

Как и в начале доказательства каждый полуинтервал Δ_i разобьем на 2^p равных частей и вновь полученные полуинтервалы обозначим через Δ_{ij}, а затем в как в (42) определим кусочно-постоянную функцию

$$\overline{u}_{\bar{k}_0+p}(t) = \left\{\hat{u}_{\bar{k}_0+p}(t) = \overline{u}_{\bar{k}_0}\left(\overline{t}_j\right) : t \in \Delta_{ij},\ i = 1,2,\ldots,2^{k_0},\ j = 1,2,\ldots,2^p\right\}. \tag{51}$$

Из (28) и (51) будет следовать, что $\hat{u}_{\bar{k}_0+p}(t) \in D\left(T_{\bar{k}_0+p}\right)$ и

$$T_{\bar{k}_0+p}\hat{u}_{\bar{k}_0+p} = v_{\bar{k}_0+p}. \tag{52}$$

Так как функция $\overline{u}_{\bar{k}_0+p}(t)$ непрерывна на отрезке $[0,1]$, то следую (51)

$$\hat{u}_{\bar{k}_0+p} \to \overline{u}_{\bar{k}_0} \text{ при } p \to \infty. \tag{53}$$

Из (53) следует, что найдется p_0 такой, что для любого $p \geq p_0$

$$\left\|\hat{u}_{\bar{k}_0+p} - \overline{u}_{\bar{k}_0}\right\| < \frac{\varepsilon}{2}. \tag{54}$$

Из (47), (48), (52) и (54) следует утверждение леммы.

Следуя (6) метод регуляризации n-го порядка приближенного решения интегрального уравнения первого рода заключается в сведении последнего к вариационной задаче

$$\inf\left\{\begin{array}{l}\int_0^1\left[\int_0^1 K(s,t)u(s)\,ds - f_\delta(t)\right]^2 dt + \alpha\int_0^1\left[\frac{d^n u(t)}{dt^n}\right]dt : \\ u,\ u^{(n)} \in L_2[0,1],\ u(0) = 0, u'(0) = 0,\ldots,u^{(n-1)}(0) = 0\end{array}\right\}. \tag{55}$$

Вариационная задача (55) разрешима единственным образом. Решение этой задачи обозначим через $u_\delta^\alpha(s)$.

Конечноразностная аппроксимация вариационной задачи (54) заключается в сведении последней к ее конечноразностному аналогу

$$\inf\left\{\frac{1}{2^k}\sum_{j=1}^{2^k}\left[\frac{1}{2^k}\sum_{j=1}^{2^k}K_{ij}u_i - f_\delta^j\right]^2 + \alpha\left\|T_k^n u\right\| : u \in D\left(T^n\right)\right\}, \tag{56}$$

где $P_k f_\delta(t) = \left\{f_\delta^j : t \in \Delta_j,\ i = 1,2,\ldots,2^k\right\}$, $u \in U_k$, $u(s) = \left\{u_i : s \in \Delta_i,\ i = 1,2,\ldots,2^k\right\}$.

Вариационная задача (56) имеет единственное решение u_δ^α такое, что $u_\delta^\alpha(k) \to u_\delta^\alpha$ при $k \to \infty$.

Заключение

В статье рассмотрен метод построения последовательности конечномерных задач, решения которых последовательно сходятся к регуляризованному решению исходной некорректно поставленной задачи. Конечная размерность построенных задач позволяет решать их при помощи компьютера. Существенным отличием от фундаментальной ра-

боты [1] является замена условий слабой замкнутости пары операторов исходной и аппроксимирующих задач на требование A и T^n одновременной полноты последовательностей $\{A_k\}$ и $\{T_k^n\}$, приближающих оператор исходной задачи и оператор стабилизатора соответственно. Выполнимость данных условий проверить значительно легче.

В дальнейшем полученные результаты могут быть применены при решении некорректных задач, возникающих в различных отраслях науки и техники.

Литература

1. Танана, В.П. Конечномерная аппроксимация метода регуляризации / В.П. Танана // Изв. вузов. Математика. — 1986. — № 6. — С. 65–69.
2. Васин, В.В. Дискретная сходимость и конечномерная аппроксимация регуляризующих алгоритмов / В.В. Васин // Журнал вычислительной математики и математической физики. — 1979. — Т. 19, вып. 1, — С. 11–21.
3. Тихонов, А.Н. О регуляризации некорректно поставленных задач. — Доклады АН СССР — 1963 — СТ.153, №1, С. 49–52.
4. Танана, В.П. Методы решения операторных уравнений / В.П. Танана — М.: Наука, 1981. — 158 с.
5. Осипов, Ю.С. Основы метода динамической регуляризации / Ю.С. Осипов, Ф.П. Васильев, М.М. Потапов — Изд-во МГУ, 1999. — 237 с.

Танана Виталий Павлович, д.ф.-м.н., профессор, зав. кафедрой вычислительной математики, Южно-Уральский государственный университет (Челябинск, Российская Федерация), tananavp@susu.ac.ru.

Бельков Сергей Игоревич, ассистент кафедры прикладной математики, Южно-Уральский государственный университет (Челябинск, Российская Федерация), sergey_belkov@mail.ru.

Поступила в редакцию 1 января 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 1, pp. 86–98

DOI: 10.14529/cmse150108

THE FINITE DIFFERENCE APPROXIMATION FOR THE TIKHONOV REGULARIZATION METHOD OF THE N-TH ORDER

V.P. Tanana, South Ural State University (Chelyabinsk, Russian Federation)
tananavp@susu.ac.ru,
S.I. Belkov, South Ural State University (Chelyabinsk, Russian Federation)
sergey_belkov@mail.ru

This article is a natural extension of the work by A.N. Tikhonov, where the idea of a finite-dimensional approximation of the regularization problem was first formulated. However, the conditions, offered for operators, are difficult to verify. In the present work we offer other conditions, which are easier to use in practice, and use it to prove the theorem of convergence of the finite-dimensional approximation for the Tikhonov regularization method. Application of the described method is demonstrated by the example with the Fredholm equation of the first kind.

Keywords: inverse problem, regularization, finite difference approximation, ill-posed problem, integral equation.

References

1. Tanana V.P. Finite-dimensional approximation of the regularization method // Soviet Math. (Iz. VUZ.). 1986. Vol. 30, No. 7. P. 89–94.
2. Vasin V.V. .Discrete convergence and finite-dimensional approximation of regularizing algorithms // U.S.S.R. Comput. Math. Math. Phys. 1979. Vol. 19, No. 1. P. 8–19.
3. Tikhonov A.N. O regulyarizatsii nekorrektno postavlennih zadach [About regularization of ill-posed problems]. Doklad AN SSSR [Reports of Academy of Sciences USSR]. 1963. Vol. 153, No. 1. P. 49–52.
4. Tanana V.P. Metody resheniya operatornih uravneniy [Methods for solving the operator equations]. Moscow, «Science», 1981. 158 p.
5. Osipov Y.S., Vasiliev F.P., Potapov M.M. Osnovy metoda dinamicheskoj regulyarizatsii [Bases of the Dynamical Regularization Method]. Moscow, Publishing of the Moscow State University, 1999. 237 p.

Received January 1, 2015.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Серия основана в 2012 году.

Свидетельство о регистрации ПИ ФС77-57377 выдано 24 марта 2014 г. Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций.

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Подписной индекс научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»: 10244, каталог «Пресса России». Периодичность выхода — 4 выпуска в год (февраль, май, август и ноябрь).

ПРАВИЛА ДЛЯ АВТОРОВ

1. Правила подготовки рукописей и пример оформления статей можно загрузить с сайта серии `http://vestnikvmi.susu.ru`. **Статьи, оформленные без соблюдения правил, к рассмотрению не принимаются и назад авторам не высылаются.**
2. Адрес редакции научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, Южно-Уральский государственный университет, факультет Вычислительной математики и информатики, кафедра СП, ответственному секретарю, доценту Цымблеру Михаилу Леонидовичу.
3. Адрес электронной почты редакции: `vestnikvmi@gmail.com`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**