ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2014
Т. 3, № 4

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

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

Основной целью издания является пропаганда научных исследований в следующих областях:

- 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

SERIES

"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2014
Vol. 3, no. 4

ISSN 2305-9052

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

South Ural State University

The main purpose of the series is publicity of scientific researches in the following areas:

- 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

Contents

УДК 004.588 DOI: 10.14529/cmse140401

МОДЕЛИ И СТАНДАРТЫ ЭЛЕКТРОННОГО ОБУЧЕНИЯ

Н.С. Силкина, Л.Б. Соколинский

Статья представляет собой обзор моделей и стандартов, используемых в современных системах электронного обучения. Описывается общая концептуальная модель среды электронного обучения. Рассматриваются: модель данных для взаимодействия с электронными образовательными объектами; модель накопления контента, определяющая структуру образовательных объектов, способы их поиска и передачи между различными обучающими системами, а также способы упаковки контента; модель среды выполнения, определяющей структуру прикладного программного интерфейса для управления образовательными объектами; модель упорядочивания иерархического образовательного контента; модель компетенций, используемая для спецификации знаний, умений и навыков в системах электронного обучения. Также дается обзор стандарта SCORM, объединяющего в себе комплекс моделей электронного обучения.

Ключевые слова: электронное обучение, e-learning, модель метаданных, модель данных, модель накопления образовательного контента, модель среды выполнения, модель простого упорядочивания, модель компетенций, SCORM.

Введение

Под электронным обучением (e-Learning) понимаются все формы обучения с помощью компьютеров, которые имеют методический характер и направлены на построение у субъекта обучения (учащегося) системы знаний с учетом его индивидуального опыта, практики и подготовки. При этом информационные и телекоммуникационные системы, прежде всего Интернет и мультимедиа, используются в качестве основной платформы реализации процесса обучения [1].

Одним из наиболее популярных подходов к электронному обучению на сегодняшний день является создание образовательных модулей на основе динамических Web-страниц. При этом имеется большая диспропорция между временем, затрачиваемым на производство образовательного модуля, и временем его использования при обучении, поскольку традиционные технологии обучения не очень просто воспроизводятся при помощи современных авторских инструментальных средств [2]. Использование единых моделей и стандартов позволит снизить стоимость разработки электронных учебных курсов за счет использования уже существующих образовательных модулей и легкости переноса образовательного контента из одной системы управления обучением в другую.

Одним из факторов популярности электронного обучения является возможность развертывания приложения электронного обучения и даже всего пакета университетского курса (учебный план, лекции, приложения регистрации, загружаемые книги, методические пособия и т.д.) на смартфонах, планшетных ПК и других подобных устройствах, что позволяет студентам по-настоящему быть мобильными, а лучшие лекции профессоров может видеть и изучать более широкая аудитория [3].

Популярность электронного обучения выросла также благодаря снижению затрат на инфраструктуру и стремлению получить качественное образование по приемлемой для студентов цене, удобству и гибкости [4].

Так прием в колледжи США на курсы электронного обучения в 2008 году был на 16,9% выше, чем в 2007-м, при общем числе 4,6 млн студентов, и более половины колледжей и

университетов в США имеют онлайн-курсы или программы электронного обучения. Открытый университет Великобритании играет важную роль в системе образования Евросоюза, наблюдается также значительный рост интереса к электронному обучению в Азии, в частности в Японии, Корее, Китае и Индии. Университетский колледж Университета Мэриленда является крупнейшим в мире образовательным учреждением, имеющим программу онлайнового образования, этот университет предлагает свыше 40 программ для получения степеней бакалавра и магистра, причем это обучение можно целиком получить в режиме онлайн [3].

Однако, на сегодняшний день отсутствуют обзоры международных стандартов и моделей в области электронного обучения как на русском, так и на английском языках. Также отсутствуют полные описания международных моделей и стандартов в области электронного обучения на русском языке.

Настоящая статья в определенной мере призвана восполнить существующий пробел.

1. Концептуальная модель среды электронного обучения

Концептуальная модель среды электронного обучения [5] представлена в виде схемы, изображенной на рис. 1. В основе модели лежит база данных (база знаний), содержащая *образовательный контент* — структурированную совокупность информации различного типа в цифровом виде (текст, графика, видео и т.д.), которая представляет собой некоторый учебный материал. Среда обучения включает в себя также специальное программное обеспечение, которое представлено в виде *системы управления обучением* (LMS) и *сервиса времени выполнения* (RTS).

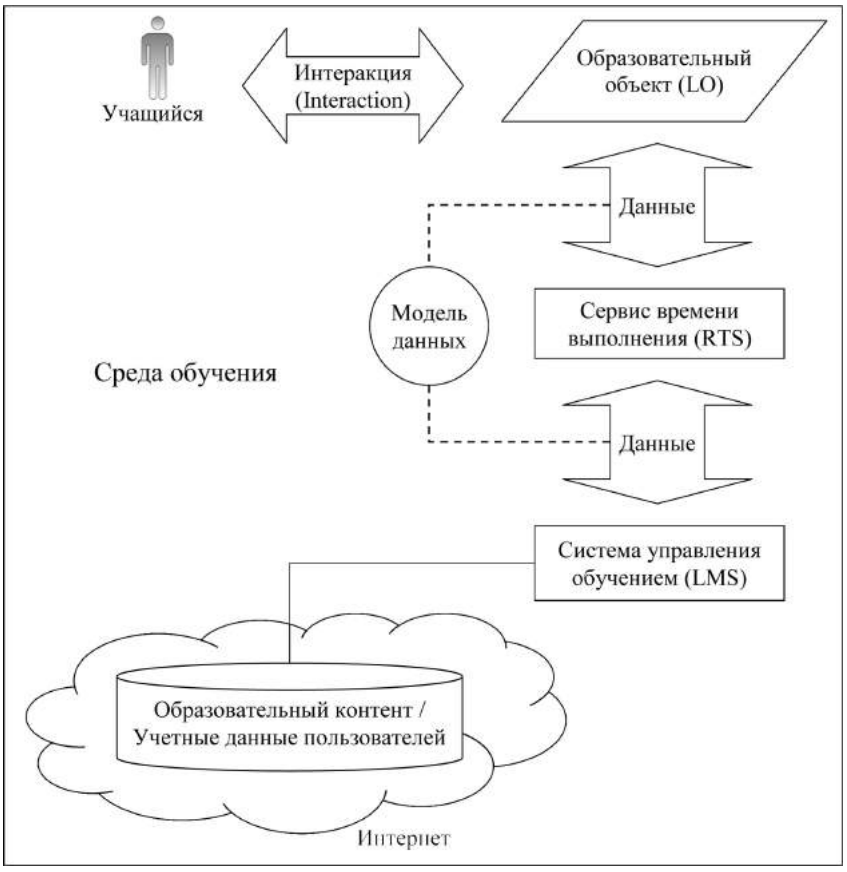

Рис. 1. Концептуальная модель среды электронного обучения

Образовательный контент, лежащий в основе схемы электронного обучения, представляет собой электронные образовательные ресурсы, распределенные в сети Интернет. Основным структурным элементом образовательного контента является *образовательный объект* (*Learning Object*) или кратко — *LO*. Объект LO представляет собой совокупность «оцифрованного» учебного материала, обладающего некоторой методической целостностью и представляющего собой единицу знания, передаваемую от LMS к учащемуся. LO обладает внутренней структурой и может включать в себя не только учебный материал, но и некоторые *инструктивные события* (*instructional events*), реализуемые в виде процедурного кода (скриптов). Например, объект LO может быть представлен в виде HTML-страницы, в которую внедрен видеоклип и скрипт ECMA, написанный в соответствии со стандартом IEEE 1484.11.2-2003 [6].

Система управления обучением (*Learning Management System*) или кратко — *LMS* представляет собой программное обеспечение, которое включает в себя функции регистрации и авторизации учащихся, администрирования образовательными объектами, управления процессом обучения, анализа результатов обучения, а также планирование и протоколирование действий учащегося.

Сервис времени выполнения (*Runtime Service*) или кратко - *RTS* представляет собой программное обеспечение, которое управляет выполнением и доставкой образовательного контента, и которое может включать в себя такие функции как распределение ресурсов, диспетчеризация процессов, управление вводом-выводом, обработка данных. Отметим, что в некоторых реализациях обучающих систем функции RTS могут интегрироваться в LMS.

Взаимодействие учащегося с обучающей системой реализуется на основе веб-протоколов и разбивается на шаги — *интеракции* (*interactions*). В ходе интеракции учащемуся доставляется образовательный объект, с которым он взаимодействует. При этом ему может демонстрироваться тот или иной учебный материал, и при определенных обстоятельствах учащийся может вводить некоторую ответную информацию в образовательный объект.

Данные, которыми обмениваются LMS, RTS и LO, должны представляться в соответствии с *моделью данных* (см. п. 3), являющейся внешней по отношению к LMS и RTS. Такой подход служит основой мобильности LO по отношению к различным системам электронного обучения. Модель данных должна поддерживать данные об учащемся (уникальный идентификатор учащегося, имя учащегося и др.), его предпочтения (уровень громкости при воспроизведении аудиоматериалов, язык обучения, скорость демонстрации образовательного контента и др.), данные об интеракции (тип), данные о взаимодействии с образовательным объектом (был ли осуществлен доступ к LO в текущем взаимодействии и по какой причине взаимодействие было прекращено: таймаут, приостановка обучения, нормальное завершение изучения LO или завершение обучения вообще), статус LO (завершено ли изучение), суммарное время изучения LO, набранные баллы и др.

2. Модель данных

Модель данных для взаимодействия с образовательными объектами (*Data Model for Content Object Communication*) была разработана Комитетом стандартизации обучающих технологий LTSC (Learning Technology Standards Committee) организации IEEE и описана в стандарте IEEE 1484.11.1-2004 [5]. Соответствующая XML-схема описана в стандарте IEEE 1484.11.3-2005 [7].

Данная модель используется объектами LO для получения от LMS данных, необходимых для их работы, а также для обратной передачи данных из LO в LMS. Структура модели представлена на рис. 2. Модель включает в себя следующие основные элементы данных:

- *comments_from_learner*: комментарии учащегося;
- *comments_from_lms*: комментарии LMS;
- *completion_status*: статус завершения;
- *completion_threshold*: граница завершения;
- *credit*: зачет;
- *entry*: вход;
- *exit*: выход;
- *interactions*: интеракции;
- *learner_id*: идентификатор учащегося;
- *learner_name*: имя учащегося;
- *max_time_allowed*: максимальное доступное время;
- *progress_measure*: степень освоения материала;
- *score*: набранный балл;
- *time_limit_action*: действие при превышении лимита времени.

```
content_object_communication: record
(
   comments_from_learner: array(0..249) of comment_type,
   comments_from_lms: array(0..99) of comment_type,
   completion_status: completion_status_type,
   completion_threshold: real(10,7) range(0..1),
   credit: state(credit, no_credit),
   data_model_version: characterstring(iso-10646-1),
   entry: state(ab_initio, resume, _nil_),
   exit: state(timeout, suspend, logout, normal, _nil_),
   interactions: bag of interaction_type,
   launch_data: characterstring(iso-10646-1),
   learner_id: long_identifier_type,
   learner_name: localized_string_type(250),
   learner_preference_data: learner_preference_type,
   lesson_status: state(passed, completed, failed,
                  incomplete, browsed, not_attempted),
   location: characterstring(iso-10646-1),
   max_time_allowed: timeinterval(second,10,2),
   mode: state(browse, normal, review),
   objectives: set of objective_type,
   progress_measure: progress_measure_type,
   raw_passing_score: real(10,7),
   scaled_passing_score: real(10,7) range(-1..1),
   score: score_type,
   session_time: timeinterval(second,10,2),
   success_status: success_status_type,
   suspend_data: characterstring(iso-10646-1),
   time_limit_action: state(exit_message, continue_message,
                  exit_no_message, continue_no_message),
   total_time: timeinterval(second,10,2),
)
```

Рис. 2. Структура модели данных

Элемент *comments_from_learner (комментарии учащегося)* представляет собой массив, состоящий из 250 элементов типа comment_type. Его значениями являются комментарии, получаемые от учащегося. В этих комментариях могут содержаться отзывы, пожелания и замечания учащегося относительно изучаемого LO.

Элемент *comments_from_lms (комментарии LMS)* представляет собой массив, состоящий из 100 элементов типа *comment_type*. Его значениями являются комментарии, которые LMS намеревается передать учащемуся, осуществляющему взаимодействие с LO. В этих комментариях может содержаться аннотация изучаемого LO или отзывы других учащихся, работавших с этим LO ранее.

Элемент *completion_status (статус завершения)* показывает, завершил ли учащийся изучение данного LO, и может принимать следующие значения:

- *completed*: учащийся освоил LO;
- *incomplete*: учащийся не в достаточной мере освоил LO;
- *not_attempted*: учащийся не предпринимал попытки изучить данный LO, либо работал с LO столь малое время, что это может быть квалифицировано как «не предпринимал попытки»;
- *unknown*: элементу не присваивалось значение.

Значение элемента *completion_status* может устанавливаться образовательным объектом, определяться LMS путем сопоставления *степени прогресса* (см. элемент *progress_measure*) с *границей завершения* (см. элемент *completion_threshold*), задаваемой инструктором на основе поставленных целей, либо вычисляться каким-то иным образом.

Элемент c*ompletion_threshold (граница завершения)* содержит вещественное значение в диапазоне от 0 до 1, с которым LMS сравнивает степень прогресса учащегося в освоении LO, для определения того, может ли образовательный объект рассматриваться как завершенный. Элемент *completion_threshold* разработан для использования в сочетании со *степенью прогресса* (см. ниже элемент *progress_measure*). Например, если *completion_threshold* для данного LO составляет 0.85 и учащийся достиг значения *progress_measure* 0.90, то *completion_status* этого LO для данного учащегося может быть установлен в состояние *completed*.

Элемент *credit (кредит)* определяет, является ли изучение данного LO обязательным для получения зачета. Он может принимать следующие значения:

- *credit*: необходим для зачета;
- *no_credit*: не является необходимым для зачета.

Элемент *entry (вход)* показывает, был ли ранее учащимся осуществлен доступ к объекту LO. Он может принимать следующие значения:

- *ab_initio*: доступа к LO не было;
- *resume*: учащийся работал с LO ранее и при выходе учащегося элемент *exit* (см. ниже) получил значение *suspend*;
- *_nil_*: информация о предшествующем доступе отсутствует.

Элемент *exit (выход)* определяет, по какой причине взаимодействие с объектом LO было прекращено. Он может принимать следующие значения:

- *timeout*: завершение по причине превышения лимита времени, задаваемого элементом *max_time_allowed* (см. ниже);
- *suspend*: учащийся временно приостановил работу с LO с намерением к нему вернуться;

- *logout*: LO сигнализировал о намерении учащегося завершить работу с обучающей средой, частью которой является этот LO;
- *normal*: нормальное завершение работы с LO;
- *_nil_*: условия выхода не определены.

Элемент данных *interactions (интеракции)* содержит информацию, имеющую отношение к тестированию или оценке знаний учащегося. Экземпляр записи *interaction_type* (см. рис. 3) в обязательном порядке должен включать в себя идентификатор интеракции. Если экземпляр записи *interaction_type* включает в себя элементы *correct_responses (правильные ответы)* или *learner_response (ответ учащегося)*, то он также должен включать в себя *type (вид тестового задания)*. Все остальные компоненты являются необязательными.

```
interactions: bag of interaction_type,
type interaction_type = record
(
   id: long_identifier_type,
   type: state( true_false, multiple_choice, fill_in,
         long_fill_in, likert, matching, performance,
         sequencing, numeric, other ),
   objectives_id: array(0..9) of long_identifier_type,
   time_stamp: date_time_type,
   correct_responses: correct_responses_type,
   weighting: real(10,7),
   learner_response: learner_response_type,
   result: choice (state (result_state, numeric) )
         of (
               result_state: state( correct, incorrect,
                     unanticipated, neutral ),
               numeric: real(10,7),
         ),
   latency: timeinterval(second,10,2),
   description: localized_string_type(250),
)
```

Рис. 3. Структура интеракций

Элемент *type (тип)* показывает вид тестового задания и может принимать следующие значения:

- *true_false*: задание вида «да/нет»;
- *multiple_choice*: задание с множественным выбором;
- *fill_in*: задание с кратким ответом (одно-два слова);
- *long_fill_in*: задание с развернутым ответом (предложение или абзац);
- *likert*: задание со шкалой ответов вида «полностью не согласен», «скорее не согласен», «нейтрален», «скорее согласен», «полностью согласен»;
- *matching*: задание на установление соответствия между элементами двух множеств;
- *performance*: пошаговое решение задачи;
- *sequencing*: задание на упорядочение элементов множества;
- *numeric*: задание с числовым ответом;
- *other*: прочие типы.

Элемент *correct_responses* описывает правильные ответы данной интеракции. Структура элемента *correct_responses* изображена на рис. 4 и представляет собой описание правильных ответов заданного вида тестового задания.

```
correct_responses: correct_responses_type,
type correct_responses_type = choice
(
   state (true_false, multiple_choice, fill_in, long_fill_in,
   likert, matching, performance, sequencing, numeric, other ),
)
of
(
   true_false: state( true, false ),
   multiple_choice: set of set of short_identifier_type,
   fill_in: bag of record
      (
         case_matters: boolean,
         order_matters: boolean,
         match_text: array(0..9) of loclized_string_type(250),
       ),
   long_fill_in: bag of record
      (
         case_matters: boolean,
         match_text: localized_string_type(4000),
      ),
   likert: short_identifier_type,
   matching: bag of bag of record
      (
         source: short_identifier_type,
         target: short_identifier_type,
      ),
   performance: bag of record
      (
         order_matters: boolean,
         answers: array(0..124) of record
           (
              step_name: short_identifier_type,
              step_answer: choice (state( literal, numeric )) of
                 (
                    literal: characterstring(iso-10646-1),
                    numeric: record
                       (
                          min: real(10,7),
                          max: real(10,7),
                        ),
                  ),
            ),
      ),
   sequencing: bag of array(0..35) of short_identifier_type,
   numeric: record
      (
          min: real(10,7),
          max: real(10,7),
       ),
   other: characterstring(iso-1046-1),
)
```

Рис. 4. Структура правильного ответа

Элемент *lerner_response* описывает ответы учащегося. Структура элемента *lerner_response* подобна структуре элемента *correct_responses*, за исключением атрибута *numeric*, который задается точным значением.

Элемент *result* описывает результат интеракции и может принимать следующие значения:

- *correct*: верный ответ;
- *incorrect*: неверный ответ;
- *unanticipated*: ответ не требовался;
- *neutral*: ответ ни правильный, ни не правильный.

Элемент *progress_measure (степень прогресса)* содержит вещественное значение в диапазоне от 0 до 1, показывающее, в какой мере учащийся продвинулся в изучении LO.

Элемент *max_time_allowed (максимальное доступное время)* содержит максимально возможное количество секунд, задающих время в течении которого учащийся может работать с LO.

Элемент *time_limit_action (действие при превышении лимита времени)* указывает LO, что нужно сделать при превышении максимального времени доступа, элемент может принимать следующие значения:

- *exit_message*: учащийся информируется о превышении времени и доступ завершается;
- *continue_message*: учащийся информируется о превышении времени, но доступ не завершается;
- *continue_no_message*: учащийся не получает сообщение о превышении времени и продолжает работать;
- *exit_no_message*: завершение доступа без вывода каких-либо сообщений.

Применение стандартизированной модели данных для обмена информацией между LMS и LO позволяет использовать без модификаций существующие объекты LO в различных системах LMS, поддерживающих данную модель.

3. Модели накопления контента

Модели накопления контента описывают компоненты, используемые в образовательных системах, способы их обмена и описания для поиска и использования, а также способы упаковки контента.

3.1. Модель метаданных

Модель метаданных образовательных объектов LOM (*Learning Object Metadata*) предназначена для описания структуры и свойств образовательных объектов LO. Обычно это описание выполняется на языке XML. Модель LOM была разработана Комитетом стандартизации обучающих технологий LTSC (Learning Technology Standards Committee) организации IEEE и описана в стандарте IEEE 1484.12.1-2002 [8]. Соответствующая XML-схема описана в стандарте IEEE 1484.12.3-2005 [9].

Основной целью LOM является обеспечение повторного использования LO, поддержка открытости и интероперабельности образовательных объектов в контексте LMS. По существу LOM описывает структуру и семантику атрибутов, которые должны быть определены при описании LO. В их число входят следующие: тип LO, автор, владелец, условия использования, формат, педагогические атрибуты и др.

Схема модели LOM изображена на рис. 5. Схема LOM представляет собой иерархию атрибутов, с помощью которых формируется мета-описание LO. Все атрибуты первого уровня делятся на девять категорий:

- *General (общая)* включает в себя атрибуты, описывающие общие свойства LO;
- *Lifecycle (жизненный цикл)* включает в себя атрибуты, описывающие текущее состояние LO и историю его изменений;
- *Meta-Metadata (мета-метаданные)* включает в себя атрибуты, относящиеся к самому экземпляру метаданных, но не к образовательному объекту, который он описывает;
- *Technical (техническая)* включает в себя атрибуты, описывающие технические требования и технические характеристики LO;
- *Educational (образовательная)* включает в себя атрибуты, определяющие образовательные и педагогические характеристики LO;
- *Rights (права)* включает в себя атрибуты, определяющие права на интеллектуальную собственность и условия использования LO;
- *Relation (связь)* включает в себя атрибуты, описывающие взаимосвязи образовательного объекта с другими образовательными объектами;
- *Annotation (аннотация)* включает в себя атрибуты, содержащие комментарии о том, каким образом следует использовать образовательный объект в процессе обучения, и информацию о том, кем и когда эти комментарии были созданы;
- *Classification (классификация)* включает в себя атрибуты, описывающие положение LO в определенной классификационной системе.

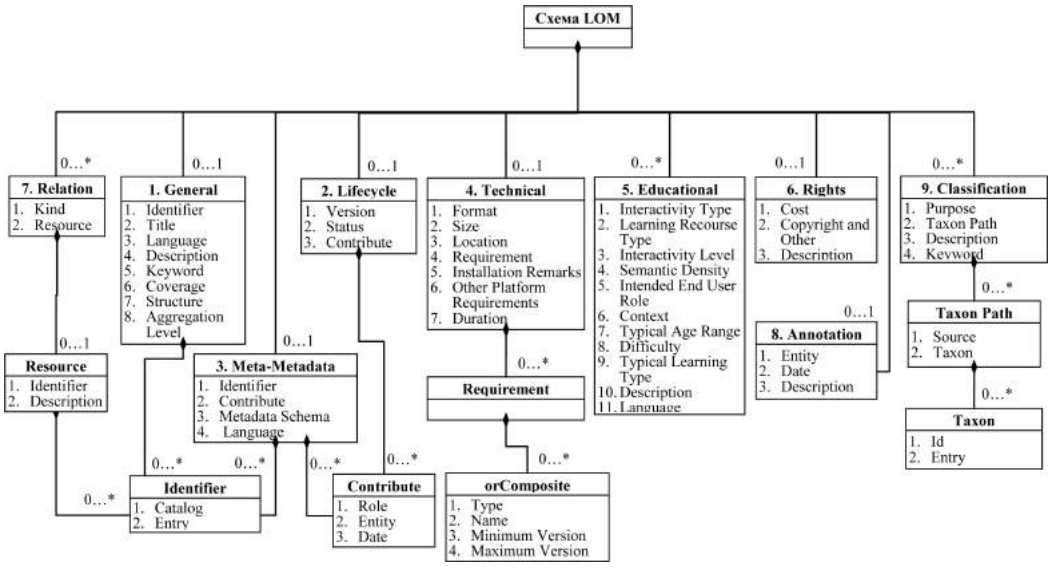

Рис. 5. Схема модели LOM

Атрибуты первого уровня могут включать в себя под-атрибуты второго уровня, которые, в свою очередь, могут включать в себя под-атрибуты третьего уровня. Семантика каждого атрибута обусловлена его положением в иерархической схеме, описывающей LO.

LOM также определяет тип данных и множество значений для каждого простого атрибута. Для многих атрибутов в качестве значения может быть введена произвольная строка символов UNICODE. Для других атрибутов вводимые значения должны выбираться из предопределенного списка или иметь специальный формат. Следующие элементы данных предусматривают специальный формат значений:

- элементы *LangString* содержат части *Language* and *String*, позволяя записывать одну и ту же информацию на разных языках;
- элементы *Vocabulary* ограничивают множество вводимых значений предопределенным списком терминов и представляют собой пару источник, значение. Источник задает определенный словарь (список терминов), а значение указывает на слово в этом словаре;
- элементы *DateTime* и *Duration* состоят из части, задающей дату или время в «машинном» формате, и части, определяющей семантику значения (например: {продолжительность в минутах, 190}).

В современном электронном обучении модель LOM является базовой для обеспечения мобильности и интероперабельности образовательных объектов.

3.2. Модель структуры контента

Модель структуры контента определяет структуру электронного учебного курса в соответствии с требованием интероперабельности. Эта модель была разработана отраслевой стандартизирующей организацией AICC (Aviation Industry Computer-based training Committee) и описана в документах CRS003 [10] и CMI001 [11]. Главная цель, которую преследовали разработчики, состояла в том, чтобы различные LMS могли бы использовать широкую гамму поставляемых электронных курсов. Дополнительным требованием была возможность сохранять структуру курса, разработанного в одной LMS, при его экспорте в другую LMS.

Схематично модель структуры контента изображена на рис. 6. В модели вводятся несколько уровней иерархии, имеющие определенные имена. Каждый уровень представляет собой определенные части учебного курса и занимает строго фиксированную позицию в иерархии. Любой элемент нижнего уровня должен являться структурной частью некоторого элемента вышестоящего уровня и не может, таким образом, превосходить его по объему. Модель наделяет уровни следующей семантикой.

Рис. 6. Модель структуры контента

Программа (Curriculum) — совокупность учебных курсов, составляющих образовательную программу.

Курс (Course) — совокупность учебного материала, обладающая дидактической целостностью, и позволяющая учащемуся освоить знания, необходимые для приобретения определенных умений и навыков.

Глава (Chapter) — значимая часть курса. Объединяет в себе несколько разделов или уроков.

Раздел (Subchapter) — значимая часть главы. Объединяет в себе несколько модулей или уроков.

Модуль (Module) — значимая часть курса, главы или раздела. Объединяет в себе несколько уроков.

Урок (Lesson) — значимая часть учебного материала, которая осваивается учащимся за одно занятие продолжительностью от 20 минут до одного часа.

Понятие (Topic) — логически самостоятельная часть урока.

Кадр (Frame) — значимый элемент графического интерфейса (окно), появляющийся на экране в определенный момент в течении урока.

Объект (Object) — компонент кадра. Объекты могут быть графическими, текстовыми или управляющими.

Электронный учебный курс имеет иерархическую структуру, включающую в себя элементы следующих трех типов:

- *Курс (Course)*;
- *Блок (Block)*;
- *Назначаемый элемент (Assignable Unit)*.

Курс представляет собой наибольшую часть учебного материала, которая может быть передана из одной системы управления обучением в другую.

Назначаемый элемент — наименьшая компонента курса, которую LMS может назначать и контролировать в процессе обучения. Обычно такой компонентой является урок.

Блок представляет собой группу назначаемых элементов и/или других блоков, логически связанных между собой. На различных уровнях иерархии блокам могут соответствовать модули, разделы или главы.

Подобная организация учебного материала обеспечивает большую гибкость при создании электронного учебного курса, поскольку позволяет описывать блоки внутри других блоков и задавать, таким образом, практически неограниченное количество уровней детализации материала.

3.3. Модель упаковки контента

Модель упаковки контента (Content Packaging Information Model) определяет стандартный набор структур данных, которые могут быть использованы для обмена образовательным контентом между различными LMS. Модель упаковки контента была разработана консорциумом IMS Global Learning Consortium и описана в спецификациях IMS [12, 13].

Общая структура модели упаковки контента изображена на рис. 7. Модель включает следующие основные структурные компоненты:

- *Логический пакет (Logical package)* представляет собой один или более *юнитов (units)* многократно используемого учебного материала. Логический пакет охватывает все множество компонент, описываемых манифестом, включая локальные компоненты и удаленные компоненты, доступные по ссылкам.
- *Пакет обмена (Interchange package)* представляет собой совокупность компонент, подлежащих обмену между различными LMS, включая манифест и другие выбранные файлы.

- *Манифест (Manifest)* представляет собой xml-документ, описывающий полный экземпляр логического пакета. Манифест может содержать ссылки на локальные или удаленные компоненты. Манифест включает в себя следующие секции:
  - *Метаданные (Metadata)* представляют собой описательную информацию о пакетах контента, логических организациях, контенте или файлах (см. п. 4.1);
  - *Организации (Organizations)* описывают логические взаимосвязи между юнитами в виде деревьев активностей (см. п. 6);
  - *Ресурсы (Resources)* представляют собой опись контента и файлов, используемых в корневом (parent) манифесте;
  - *Под-манифесты (Child-manifests)* представляют собой полноценные подчиненные манифесты, которые присутствуют или на которые имеется ссылка в корневом манифесте. Каждый под-манифест описывает полноценный логический пакет, являющийся частью объемлющего логического пакета. Под-манифест может являться корневым для под-манифеста еще более низкого уровня. Иерархия манифестов отражает иерархическую организацию образовательного контента.
- *Файлы (Files)* воплощают в себе образовательный контент, описываемый логическим пакетом.

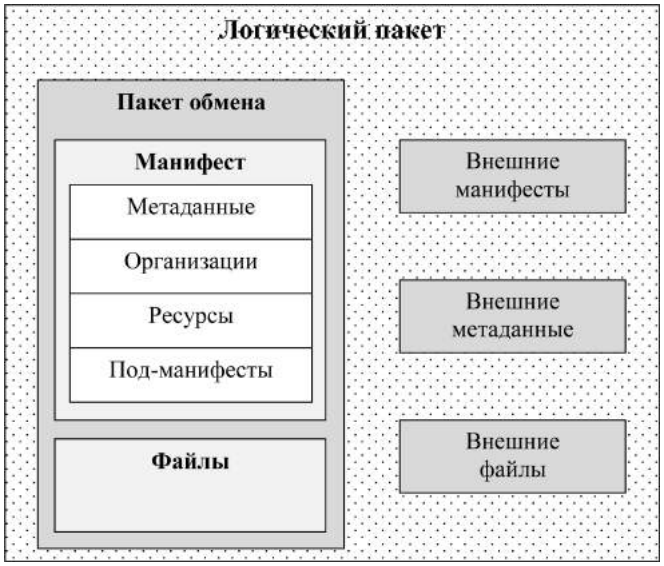

Рис. 7. Структура модели упаковки контента

Любой из описанных компонентов пакета обмена может быть внешним по отношению к нему. В этом случае пакет обмена содержит только ссылку на удаленный компонент. Пакет обмена вместе с удаленными компонентами и образует логический пакет. Детальное описание компонентов и соответствующие xml-схемы можно найти в [13].

4. Модель среды выполнения

Модель среды выполнения (Runtime Environment Model) определяет структуру *прикладного программного интерфейса (Application Program Interface)* или кратко *API*, используемого для организации взаимодействия между образовательным объектом (LO) и сервисом времени выполнения (RTS). Модель среды выполнения была разработана Комитетом стандартизации обучающих технологий LTSC (Learning Technology Standards Committee) организации IEEE и описана в стандарте IEEE 1484.11.2-2003 [6]. Основной целью, преследу-

емой при разработке этой модели, было обеспечение интероперабельности образовательных объектов в различных средах выполнения.

Схематично модель среды выполнения изображена на рис. 8. Общепринятой реализацией этой модели является программная среда, ориентированная на доставку контента посредством web-обозревателя, в которой все коммуникации инициируются образовательным объектом LO. Такая коммуникационная модель не предусматривает возможности коммуникаций между RTS и LO, инициированных со стороны RTS. Интероперабельность достигается путем посылки запросов через общие коммуникационные методы, реализованные в API. Организация коммуникаций и интерфейс методов являются одинаковыми для каждого экземпляра реализации API.

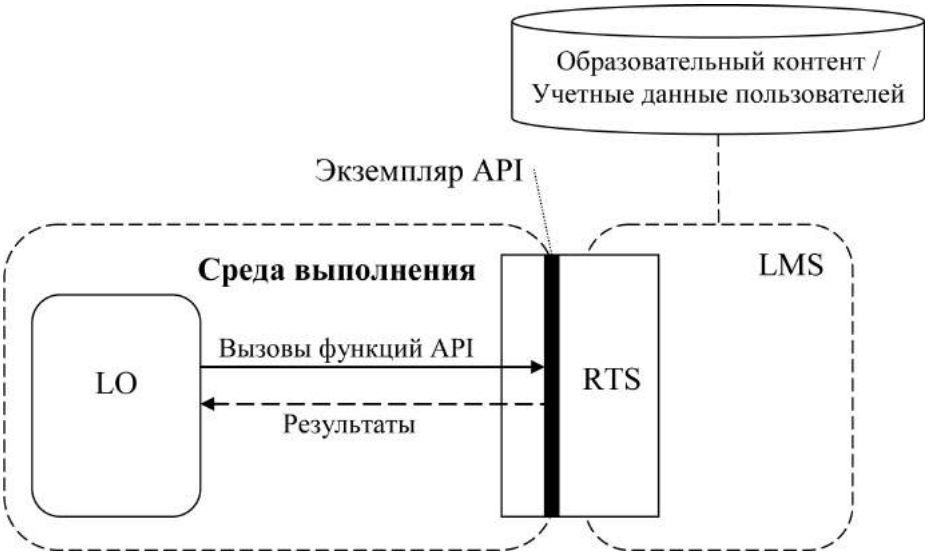

Рис. 8. Модель среды выполнения

Как показано на рис. 8, RTS реализует API в среде выполнения LO. При разработке LO разработчик включает в его реализацию средства обнаружения и коммуникации с экземпляром API. LMS или ее клиент, обеспечивающие доступ к репозиторию с образовательным контентом (локальному или удаленному), предоставляет для учащегося RTS. RTS либо доставляет учащемуся LO и осуществляет его запуск, либо загружает URI (Uniform Resource Identifier) для активизации LO посредством web-обозревателя. Сразу после своего запуска LO осуществляет поиск экземпляра API. Как только экземпляр API найден, LO инициирует сеанс связи с RTS. В ходе сеанса связи образовательный объект может запрашивать необходимые ему данные через экземпляр API. Через этот же экземпляр API RTS возвращает затребованные данные или сообщение об ошибке. В ходе сеанса связи LO может пересылать или устанавливать элементы модели данных (см. п. 3) для сохранения в репозитории. RTS может использовать элементы данных или другую сохраненную информацию в отчетах о статусе учащегося по отношению к данному образовательному объекту. В свою очередь, LO может получить детализированное сообщение об ошибке.

Образовательный объект может продолжать подобные коммуникации, запрашивая и передавая данные до момента, когда учащийся завершит изучение LO, либо учащийся прервет сеанс связи до окончания изучения LO, либо сеанс связи будет завершен аварийно (отказ системы, потеря питания или др.). В первых двух случаях LO явно сообщает экземпляру API о закрытии сеанса связи. В последнем случае, RTS не получает сигнала через экземпляр API о закрытии сеанса связи. RTS необходимо самостоятельно обработать ис-

ключительную ситуацию в соответствии со спецификациями экземпляра API. На рис. 9 представлена общая структура API.

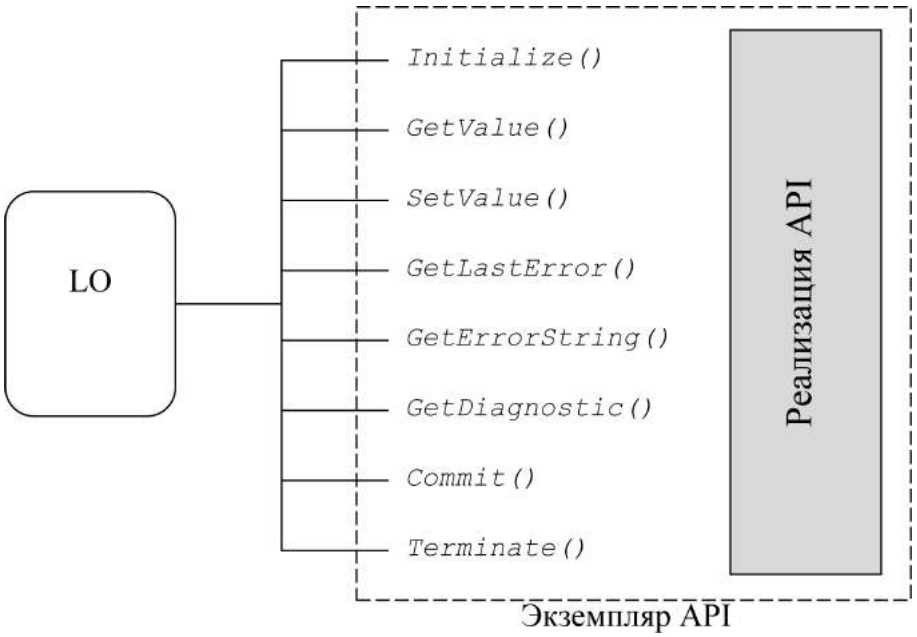

Рис. 9. Структура API

5. Модель простого упорядочения

Модель простого упорядочения (Simple Sequencing Information and Behavior Model) разработана консорциумом IMS Global Learning Consortium и описана в спецификациях IMS [14–16]. Данная модель определяет метод, с помощью которого, исходя из имеющегося преподавательского опыта, можно задать план изучения образовательного контента, на основе которого LMS упорядочивает отдельные *обучающие активности (learning activity)*, непротиворечивым образом. Разработчик образовательного контента задает относительный порядок предоставления учащемуся образовательных объектов и условия, при которых части образовательного контента выбираются, доставляются или пропускаются в процессе обучения.

Модель простого упорядочения определяет поведение и функциональность, которые должны поддерживаться стандартной LMS. Она включает правила, формирующие поток активностей в контексте определенного образовательного контента, в зависимости от результатов взаимодействия учащегося с этим контентом. Такое представление планируемого инструктивного потока может быть создано вручную или с помощью авторинговых систем, генерирующих результат, в соответствии со спецификациями данной модели. Созданное представление упорядочивания может использоваться различными системами, разработанными для доставки инструктивных активностей учащемуся.

Простое упорядочение обозначено как *простое* потому, что оно включает ограниченный набор наиболее распространенных стратегий упорядочивания, а не потому, что спецификации этой модели являются простыми. Простое упорядочение не охватывает все возможные аспекты, связанные с проблемой упорядочивания учебного материала в LMS. В частности, простое упорядочение не включает в себя, но и не запрещает использовать: упорядочивание с использованием искусственного интеллекта; упорядочивание на основе плана заня-

тий; упорядочение, требующее данные от внешних закрытых систем и сервисов (например, упорядочивание на базе встроенной имитационной модели); коллаборативное (совместное) обучение; индивидуальное обучение; синхронизацию между параллельными обучающими активностями и др.

Модель простого упорядочения опирается на понятие обучающей активности. *Обучающая активность (learning activity)* — это отдельное задание (instructional event), являющееся объектом упорядочивания. Процесс взаимодействия учащегося с активностью делится на последовательные во времени интервалы, называемые *попытками*. Попытка считается *успешной*, если учащийся выполнил все шаги задания, представленного в данной активности. Учащийся может завершить попытку досрочно, не выполнив всех шагов задания, и начать следующую попытку. Выполнение попытки может прерываться на некоторое время, в течение которого активность будет находиться в *приостановленном* состоянии. Впоследствии учащийся может возобновить прерванную попытку. Таким образом, одна попытка может распространяться на несколько сеансов работы учащегося с LMS. При анализе результатов взаимодействия учащегося с активностью, система учитывает количество попыток, их успешность, продолжительность взаимодействия с активностью и абсолютную продолжительность активности (см. рис. 10).

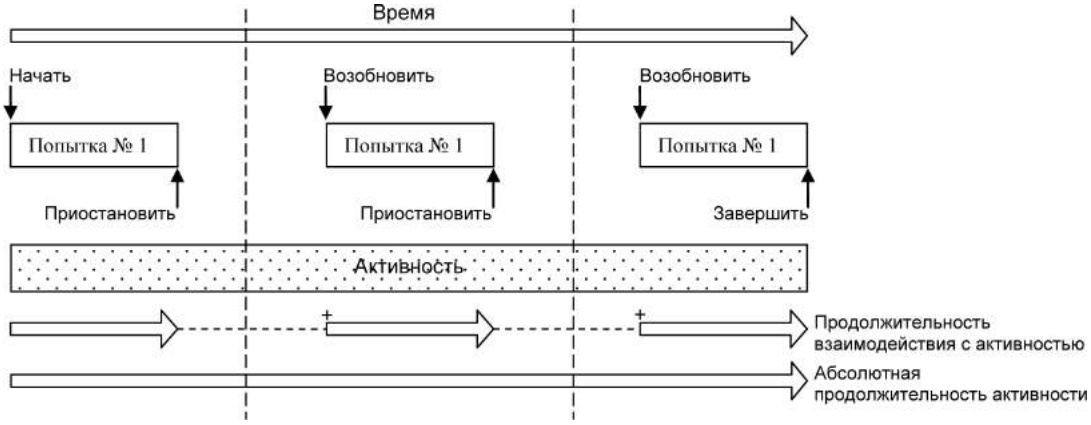

Рис. 10. Временная диаграмма взаимодействия с активностью

Отдельные активности могут группироваться для формирования активности более высокого уровня и включать в себя несколько уровней подчиненных активностей, образуя *дерево активностей (activity tree)*. Пример дерева активностей приведен на рис. 11. Здесь, например, активность AA является частью активности A, а активность AABA является частью активности AAB, которая является частью активности AA.

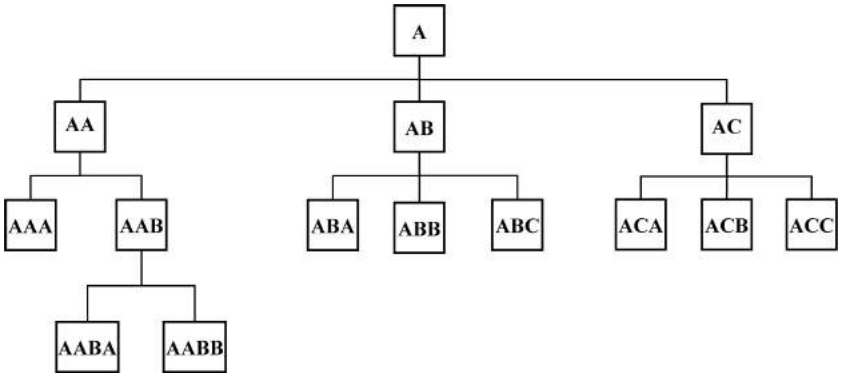

Рис. 11. Дерево активностей

Различным уровням иерархии в дереве активностей могут соответствовать различные концептуальные метки. Например, в рамках одного учебного курса активности A соответствует курс в целом, AA является уроком, AAB — этапом урока, а AABB — шагом этапа. С листовыми активностями связываются образовательные ресурсы, представляющие собой контент, который должен быть доставлен учащемуся.

В качестве канонического порядка обхода дерева активностей модель простого упорядочения предполагает обход дерева в прямом порядке (pre-order traversal). В соответствии с указанным порядком сначала нужно посетить корень, а затем слева-направо рекурсивно совершить обход всех поддеревьев. Для примера на рис. 11 порядок обхода узлов в этом случае будет следующим: A, AA, AAA, AAB, AABA, AABB, AB, ABA, ABB, ABC, AC, ACA, ACB, ACC. Маршрут обхода дерева по умолчанию может быть изменен с помощью *правил упорядочивания (sequencing rules)*, созданных разработчиком образовательного контента. Обход дерева начинается при поступлении запроса упорядочивания, который, в свою очередь, инициируется учащимся с помощью навигационных событий, либо системой доставки контента самостоятельно. Правила упорядочивания вычисляются во время выполнения и могут зависеть от результатов предшествующей работы учащегося. В каждый момент времени учащемуся всегда доставляется только одна активность, с которой может быть связано произвольное количество образовательных ресурсов.

Правила упорядочивания ассоциируются с кластерами узлов в дереве активностей. Под *кластером* в модели простого упорядочения понимается выделенный узел и все его сыновья (см. рис. 12). Действия правила упорядочивания никогда не выходят за пределы кластера. По умолчанию, учащийся может выбрать любую дочернюю активность в кластере, однако, мы также можем задать режим *рекомендуемого маршрута (guided flow)*, при котором учащийся будет совершать обход кластера в прямом порядке, как это показано на рис. 13. Правило, которое задает такую возможность, связывается с родительским узлом кластера. Если дочерние активности в кластере являются листьями, с которыми ассоциированы образовательные ресурсы, они будут доставляться учащемуся последовательно в указанном порядке.

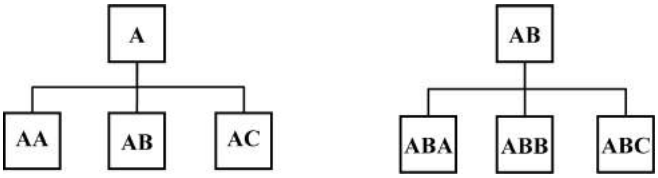

Рис. 12. Примеры кластеров

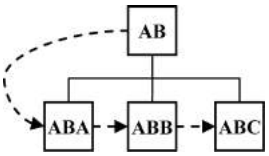

Рис. 13. Обход кластера в прямом порядке

Основными управляющими режимами, определяющими порядок обхода кластера, являются *выбор (choice)* и *маршрут (flow)*, каждый из которых может быть активирован или деактивирован. Если режим выбор деактивирован, то учащийся не может выбирать произвольный порядок обхода кластера, но должен следовать рекомендуемому маршруту.

Если режим маршрут деактивирован, то учащийся должен сам определять порядок обхода кластера. Оба режима могут быть активированы одновременно, в этом случае учащийся может выбирать активности произвольно или предпочесть рекомендуемый маршрут. Однако оба режима не могут быть одновременно деактивированы. Для всех кластеров в дереве активностей могут быть специфицированы одинаковые управляющие режимы, либо они могут отличаться у разных кластеров.

Более сложное навигационное поведение может быть организовано с помощью *правил упорядочивания (sequencing rules)*, назначаемых различным узлам дерева активностей на различных уровнях иерархии. Каждое такое правило задает условия, при которых ассоциированный с ним узел должен быть пропущен. На рис. 14 приведен пример обхода дерева активностей при активированном режиме маршрут. Штриховкой помечены узлы, для которых «сработало» правило «пропустить узел».

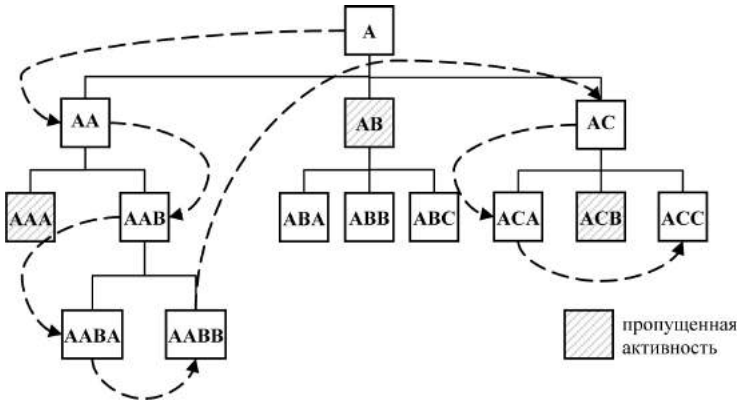

Рис. 14. Пример обхода дерева активностей с пропуском узлов

Многие правила упорядочивания, определенные в модели простого упорядочения, базируются на данных, полученных путем отслеживания результатов взаимодействия учащегося с активностями. С каждой активностью в дереве активностей связывается группа *атрибутов мониторинга*, которая делится на две подгруппы: 1) *атрибуты цели*, содержащие информацию о степени достижения цели обучения; 2) *атрибуты взаимодействия*, содержащие количественную информацию о действиях учащегося по отношению к данной активности.

Примерами атрибутов цели являются: *статус достижения цели (objective satisfied status)* — логический атрибут, показывающий, достигнута ли дидактическая цель, предусмотренная для данной активности, или нет; *нормализованная мера достижения цели (objective normalized measure)* — вещественный атрибут, принимающий значения из отрезка [-1, 1], который показывает в какой мере учащийся приблизился к достижению цели, ассоциированной с данной активностью.

Примерами атрибутов взаимодействия являются: *продолжительность взаимодействия с активностью (activity experienced duration)* — атрибут, показывающий совокупное время взаимодействия учащегося с активностью без учета времени, когда активность находилась в приостановленном состоянии (см. рис. 10); *абсолютная продолжительность активности (activity absolute duration)* — атрибут, показывающий совокупное астрономическое время взаимодействия учащегося с активностью с учетом времени, когда активность находилась в приостановленном состоянии (см. рис. 10); *количество попыток (activity attempt count)* — атрибут, показывающий количество попыток выполнения активности (нулевое значение атрибута означает, что активность еще ни разу не была запущена учащимся,

положительное значение означает, что активность находится в процессе выполнения, либо завершена учащимся); *статус завершения попытки (attempt completion status)* — логический атрибут, показывающий, была ли завершена попытка выполнения активности; *мера завершения попытки (attempt completion amount)* — вещественный атрибут, принимающий значения из отрезка [0, 1], который показывает, как далеко учащийся находится от завершения попытки.

Атрибуты мониторинга родительского узла могут вычисляться на основе анализа значений атрибутов мониторинга дочерних активностей. Это достигается путем использования *правил мониторинга (rollup rulles)*. На рис. 15 приведен пример правила мониторинга, которое вычисляет значение атрибута *статус достижения цели* для активности AB на основе значений атрибутов *статус достижения* цели дочерних активностей кластера. При этом некоторые дочерние активности кластера могут иметь пометку «не учитывать», что позволяет не учитывать значения атрибутов мониторинга данных активностей при вычислении правил мониторинга.

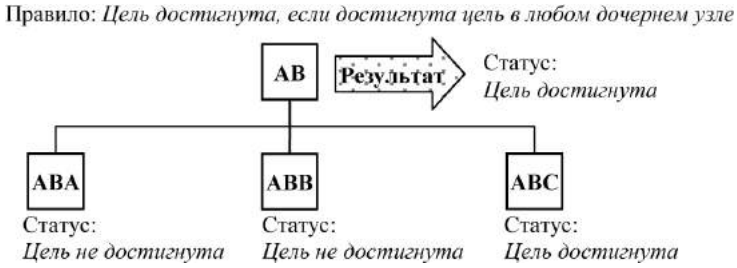

Рис. 15. Пример правила мониторинга для расчета статуса активности

Все LMS, реализующие модель простого упорядочения, выполняют следующую последовательность действий, называемую *процессом простого упорядочивания (simple sequencing process)*.

Начало сессии упорядочивания

1. Учащийся инициирует сеанс работы с LMS и устанавливает контекст, выбирая конкретный модуль электронного учебного курса.
2. LMS инициирует *сессию упорядочивания* с помощью навигационного запроса *«Start (Начать)», «Resume All (Возобновить все)»* или *«Choice (Выбрать)»*.
3. *Блок навигации (navigation behavior)* транслирует навигационный запрос «Start», «Resume All» или «Choice» в соответствующий запрос упорядочивания и выполняет его. Сессия упорядочивания «официально» начинается, когда найдена активность для предоставления учащемуся (успешно завершены шаги 4 и 5).

Цикл упорядочивания

4. Используя значения атрибутов мониторинга, *блок упорядочивания (sequencing behavior)* выполняет обход дерева активностей и определяет, какая активность (узел в дереве активностей) должна быть доставлена учащемуся на данном шаге.
5. *Блок доставки (delivery behavior)* определяет, могут ли быть доставлены учащемуся образовательные ресурсы, ассоциированные с выбранной активностью: активность должна быть листовой; предыдущая попытка взаимодействия с активностью должна быть завершена; активность не должна быть *заблокированной (disabled)*, и, в случае положительного ответа, подготавливает для доставки необходимые образовательные ресурсы.

Если выбранная активность не может быть предоставлена учащемуся, тогда общий процесс упорядочивания останавливается и ожидает следующего навигационного запроса (шаг 8).

6. Процесс упорядочивания простаивает, ожидая навигационных запросов пока учащийся взаимодействует с образовательным ресурсом.
7. Во время взаимодействия учащегося с образовательным ресурсом активность может передавать значения, которые используются для обновления значений атрибутов мониторинга.
8. Учащийся, LMS или активность инициирует навигационное событие, такое как *«Continue (продолжить)»* , *«Previous (предыдущий)»*, *«Choose activity X (выбрать активность X)»*, *«Abandon (отказаться)»*, *«Exit (выйти)»* и др.
9. LMS информирует процесс упорядочивания о произошедшем навигационном событии путем формирования навигационного запроса.
10. Если навигационный запрос показывает, что учащийся хочет завершить сессию упорядочивания блок навигации транслирует навигационный запрос в *запрос завершения (termination request)*, и сессия упорядочивания завершается. В противном случае блок навигации транслирует навигационный запрос в запрос упорядочивания.
11. Если навигационный запрос порожден в результате нормального или досрочного завершения активности, активность может обновить значения определенных атрибутов мониторинга, после чего активность завершает свою работу. Далее LMS активизирует *блок учета (rollup behavior)* для анализа всех изменений, произошедших в результате взаимодействия учащегося с активностью. Блок учета обновляет значения атрибутов мониторинга данной активности и всех ее предков в дереве активностей.
12. Процесс повторяется, начиная с шага 4, до тех пор, пока сессия упорядочивания не завершится.

Схематично процесс простого упорядочивания изображен на рис. 16.

6. Стандарт SCORM

Стандарт *SCORM (Sharable Content Object Reference Model)* [17–20] объединяет в себе модели, которые были описаны в разделах 3-6. SCORM описывает организацию и структуру образовательного контента и систему управления обучением, позволяет обеспечить совместимость образовательных объектов (LO) и возможность их многократного использования. Образовательные объекты SCORM могут включаться в разные учебные курсы и использоваться разными LMS, поддерживающими стандарт SCORM, независимо от того, кем, где и с помощью каких средств они были созданы.

На рис. 17 изображены компоненты образовательного контента SCORM от самых маленьких (assets) до самых крупных (учебных планов).

Образовательные объекты в SCORM могут быть двух типов: *asset (образовательный элемент)* и *разделяемый объект контента SCO (Shareable content object)*.

Объект *asset* — это электронное представление аудиовизуальных средств, используемых при обучении. В качестве объекта asset могут фигурировать текст, картинка, веб-страница, аудио- или видеоматериал и др. Объекты asset не могут взаимодействовать с LMS напрямую. Объекты asset могут быть многократно использованы как в различных контекстах одного электронного учебного курса, так и в разных электронных учебных курсах.

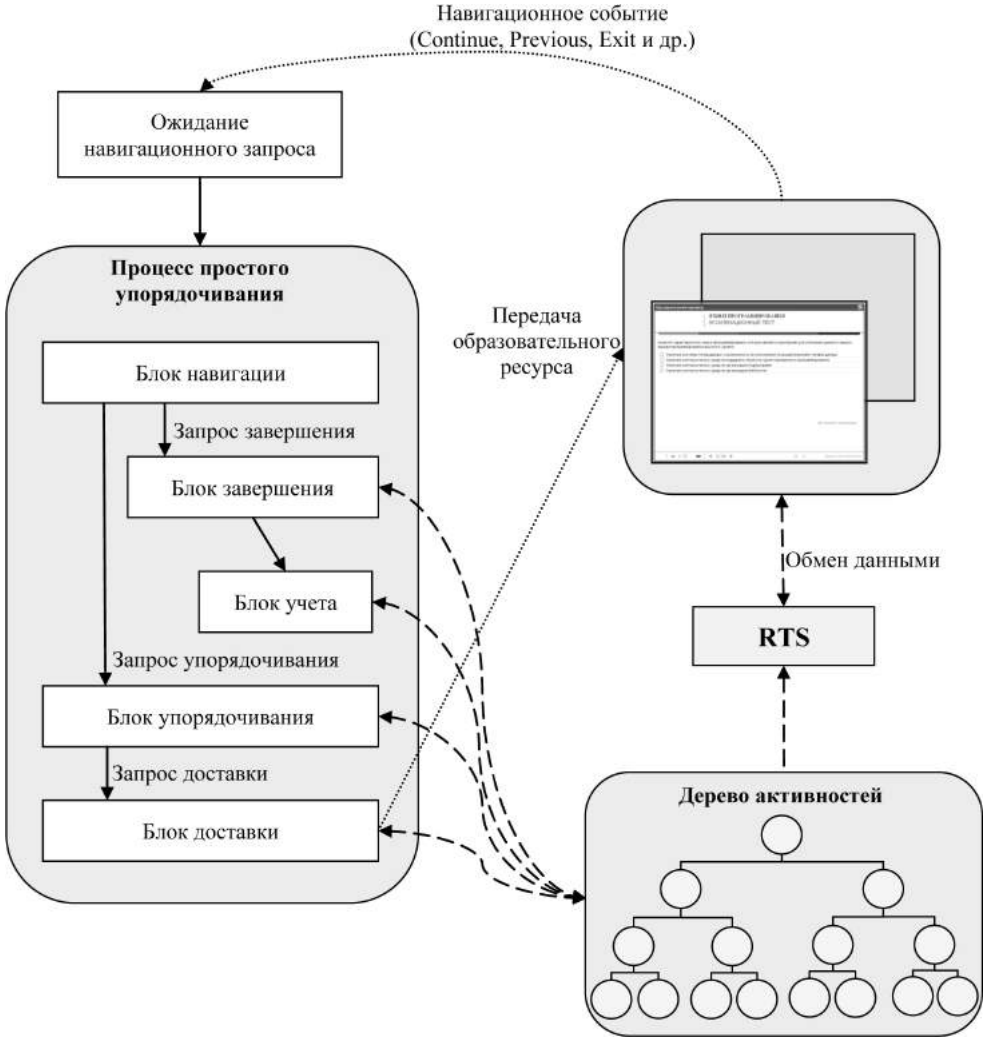

Рис. 16. Цикл упорядочивания

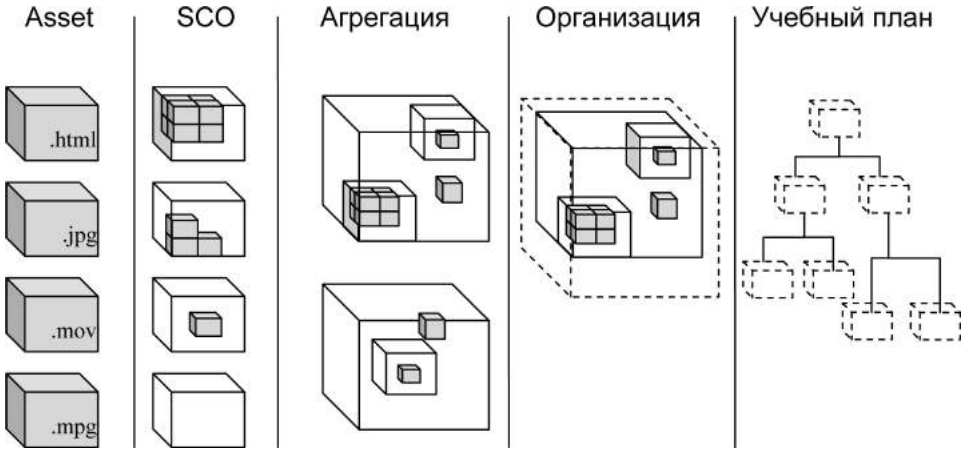

Рис. 17. Компоненты образовательного контента в SCORM

Разделяемый объект контента SCO представляет собой коллекцию из одного или более объектов asset и является наименьшей логической единицей обучения, которая может быть доставлена учащемуся с помощью LMS. С точки зрения реализации SCO является веб-приложением, которое взаимодействует с LMS посредством стандартного API (см. п. 5). SCO может получать от LMS и сохранять в LMS значения параметров мониторинга

(см. п. 6). Например, он может сохранить в LMS, такие значения, как *оценка* или *статус завершения*, или получить от LMS, такие значения, как *имя учащегося*.

Агрегация (aggregation) — это коллекция связанных обучающих активностей. Агрегация может включать в себя объекты SCO и вложенные агрегации. Агрегация соответствует кластеру в дереве активностей, описываемому в модели простого упорядочения (см. п. 6). С точки зрения реализации агрегация — это структура, описываемая в *манифесте SCORM*, где правила упорядочивания применяются к коллекции логически связанных объектов SCO или вложенных агрегаций. Структура манифеста SCORM основана на подходе, описанном в разделе 4.3. На рис. 18 изображен пример агрегации «Методы сортировки», которая включает в себя шесть вложенный объектов SCO. Каждый из вложенных объектов SCO также может представлять собой агрегацию.

Рис. 18. Пример агрегации

Организация (organization) — это раздел пакета контента, описываемого моделью упаковки контента (см. п. 4.3), в котором определяются логические связи между образовательными объектами для формирования древовидной структуры. Организация вцелом описывает структуру образовательного контента, которую автор намерен поставлять в виде единого пакета контента. Каждая организация является высокоуровневой агрегацией и рассматривается как корень агрегации. Изначально модель упаковки контента предусматривала возможность включать в один пакет несколько организаций, однако в действующей версии стандарта SCORM эта возможность не поддерживается, и в пакет можно включать только одну организацию. Организация контента является аналогом дерева активностей, описываемого в разделе 6.

Учебный план (curriculum) представляет собой древовидную структуру, узлами которой могут являться независимые компоненты. В качестве таких компонент могут фигурировать курсы, уроки, тесты и др. Для их представления могут использоваться организации SCORM. Описание структуры учебного плана выходит за рамки стандарта SCORM.

В SCORM весь образовательный контент в конце концов представляется в виде пакета контента. Пакет контента может включать единственный объект SCO или состоять из сотен таких объектов. Это зависит от разработчика образовательного контента и цели его создания. Пакет контента SCORM представляет собой zip-файл, который включает в себя все необходимое для доставки образовательного контента учащемуся, а именно манифест SCORM и файлы, в которых хранятся образовательные объекты SCO и asset, описываемые в файле манифеста. *Манифест SCORM* представляет собой xml-файл, содержащий описания всех объектов SCO и asset, которые включены в пакет контента, описания логических взаимосвязей между образовательными объектами, описания правил упорядочивания и метаданных. Как уже указывалось выше, структура манифеста SCORM основана на подходе, описанном в разделе 4.3.

Метаданные — это информация, которая описывает, чем является образовательный контент. Метаданные описывают отдельные образовательные объекты (asset и SCO) и па-

кеты контента. Метаданные позволяют разработчикам курсов осуществлять поиск контента или образовательных объектов и определять, будут ли эти объекты полезны, прежде чем они будут загружены или разработчик курса запросит права доступа. Метаданные SCORM основаны на модели метаданных, описанной в разделе 4.1.

Стандарт SCORM позволяет разработчику курса задавать порядок изучения образовательных объектов. Данный механизм может предоставлять учащемуся некоторую свободу при выборе образовательного объекта для изучения либо может жестко ограничивать этот порядок. Подробнее данный механизм описан в разделе 6.

Распространенным шаблоном проектирования электронных учебных курсов является наличие обязательного условия для некоторой обучающей активности. Когда доступность одной обучающей активности зависит от некоторых внешних условий, таких как степень завершения набора активностей или степень достижения цели набора активностей. Например, на рис. 19 показана блок-схема с серией уроков и последующего контрольного теста. Тест является заблокированным для учащегося до тех пор, пока Урок 1 и Урок 2 не будут им завершены.

Рис. 19. Схема обучения из двух лекций, за которыми следует контрольный тест

Для реализации такого шаблона необходимо создать агрегацию уроков (кластер), добавить правило мониторинга, задать цели и создать предусловие для прохождения теста (см. раздел 6). Пример реализации кластера представлен на рис. 20. Кластер включает в себя

```
<item identifier="CLUSTER-ONE" isvisible="true" >
  <title>Lesson Aggregation</title>
  <item identifier="LESSON1" identifierref="RES-SCO1"
                    isvisible="true">
    <title>Child 1</title>
  </item>
  <item identifier="LESSON2" identifierref="RES-SCO2"
                    isvisible="true">
    <title>Child 2</title>
  </item>
  <item identifier="ASSESSMENT" identifierref="RES-SCO3"
                    isvisible="true">
    <title>Child 3</title>
    <!-- Pre-Condition Rule on Assessment (реализация представлена
на рис. 23)-->
  </item>
  <imsss:sequencing>
    <imsss:controlMode flow="true" choice="true"/>
    <!--Rollups: (реализация представлена на рис. 22)
             1)If any are incomplete, the cluster is incomplete
             2)If all are completed, the cluster is complete-->
    <!-- Map Aggregation's Primary Objective (реализация
представлена на рис. 21)>
```

Рис. 20. Пример реализации кластера

три дочерних элемента (у третьего элемента задано предусловие), описание правил мониторинга и описание целей. Правила мониторинга описывают следующие условия: «если хотя бы один из уроков не пройден, тогда кластер не завершен» и «если все уроки пройдены,

тогда кластер завершен». Примеры реализации цели, правил мониторинга и предусловия представлены на рис. 21, рис. 22 и рис. 23, соответственно.

```
<!-- Map Aggregation's Primary Objective >
<adlseq:objectives>
  <adlseq:objective objectiveID = "obj-primary">
    <adlseq:mapInfo targetObjectiveID = "obj-global-lessons"
                    writeCompletionStatus="true" />
  </adlseq:objective>
</adlseq:objectives>
```

Рис. 21. Пример реализации целей

```
<!--Rollups: 1)If any are incomplete, the cluster is incomplete
             2)If all are completed, the cluster is complete-->
<imsss:rollupRules rollupObjectiveSatisfied="true"
                   rollupProgressCompletion="true">
  <imsss:rollupRule childActivitySet="any">
    <imsss:rollupConditions conditionCombination="any">
      <imsss:rollupCondition operator="not" condition="completed"/>
    <imsss:rollupConditions>
      <imsss:rollupAction action="incomplete"/>
  </imsss:rollupRule>
  <imsss:rollupRule childActivitySet="all">
    <imsss:rollupConditions conditionCombination="any">
      <imsss:rollupCondition condition="completed"/>
    </imsss:rollupConditions>
    <imsss:rollupAction action="complete"/>
  </imsss:rollupRule>
</imsss:rollupRules>
```

Рис. 22. Пример реализации правил мониторинга

```
<!-- Pre-Condition Rule on Assessment -->
<imsss:sequencing>
 <imsss:sequencingRules>
  <imsss:preConditionRule>
    <imsss:ruleConditions conditionCombination="any">
      <imsss:ruleCondition operator="not" condition="completed"
           referencedObjective="obj-prereqs" />
      <imsss:ruleCondition operator="not"
           condition="activityProgressKnown"
           referencedObjective="obj-prereqs" />
    </imsss:ruleConditions>
    <imsss:ruleAction action="disabled" />
  </imsss:preConditionRule>
 </imsss:sequencingRules>
<imsss:rollupRules rollupObjectiveSatisfied="true"
           rollupProgressCompletion="true" />
 <imsss:objectives>
  <imsss:primaryObjective objectiveID="obj-primary"
           satisfiedByMeasure="false"/>
  <imsss:objective objectiveID="obj-prereqs"
           satisfiedByMeasure="false">
    <imsss:mapInfo targetObjectiveID="obj-global-lessons"
           readSatisfiedStatus="true"/>
  </imsss:objective>
 </imsss:objectives>
<imsss:deliveryControls objectiveSetByContent = "true"/>
<adlseq:objectives>
 <adlseq:objective objectiveID = "obj-prereqs">
    <adlseq:mapInfo targetObjectiveID = "obj-global-lessons"
           readCompletionStatus="true" />
  </adlseq:objective>
 </adlseq:objectives>
</imsss:sequencing>
```

Рис. 23. Пример реализации предусловия

Более детальные сведения о разработке учебных курсов на базе стандарта SCORM можно найти в [19].

7. Модель компетенций

Модель компетенций используется для спецификации *знаний, умений и навыков (ЗУНов)* в LMS и в компетентностных профилях учащихся. Модель компетенций предоставляет средства для общепонятного описания компетенций, являющихся составной частью учебного плана, для спецификаций ЗУНов, необходимых для начала обучения (learning prerequisites), и ЗУНов, получаемых в результате обучения (learning outcomes). Модель компетенций может использоваться для обмена ЗУНами между различными LMS, системами управления персоналом (human resource management system), образовательным контентом, хранилищами компетенций и другими аналогичными системами [21].

Модель компетенций была разработана Комитетом стандартизации обучающих технологий LTSC (Learning Technology Standards Committee) организации IEEE и описана в стандартах IEEE 1484.20.1-2007 [22] и IEEE SRCM [23]. Стандарт IEEE 1484.20.1-2007 содержит формальное описание *структуры определения компетенции*. Стандарт IEEE SRCM определяет информационную модель для представления связей между различными компетенциями в виде ориентированного ациклического графа, называемого *графом компетенций*.

Структура определения компетенции изображена на рис. 24 и включает в себя следующие основные атрибуты:

- *identifier*: идентификатор компетенции;
- *title*: название компетенции;
- *description*: описание компетенции;
- *definition*: определение компетенции.

Identifier представляет собой глобальный идентификатор компетенции, который должен быть уникален во всех системах, хранящих или обрабатывающих данную компетенцию.

Title представляет собой понятное для человека название компетенции. Название может быть продублировано на нескольких языках. В качестве названия компетенции, например, может быть текстовая строка «Знание английского языка».

Description представляет собой понятное для человека описание компетенции, которое предназначено только для интерпретации человеком. Описание может быть продублировано на нескольких языках. В качестве описания компетенции, например, может быть текстовая строка «Владение письменным и устным английским языком».

Definition представляет собой структурированное описание компетенции, которое включает в себя непустой набор *утверждений (statements)*.

Statement (утверждение) описывает отдельное свойство компетенции и включает в себя следующие элементы:

- *statement_id*: идентификатор утверждения,
- *statement_name*: имя утверждения,
- *statement_text*: описание утверждения,
- *statement_token*: лексема.

Statement_id представляет собой уникальную метку в рамках данного определения.

Statement_name представляет собой имя утверждения, которое должно быть уникальным в рамках данного определения. В качестве имени утверждения может быть исполь-

```
reusable_competency_definition: record
(
   identifier: long_identifier_type,
   title: bag of langstring_type(1000),
   description: bag of langstring_type(4000),
   definition: definition_type,
   metadata: metadata_type,
)

definition_type = record
(
   model_source: characterstring(iso-10646),
   statement: statement_type,
)

statement_type = record
(
   statement_id: long_identifier_type,
   statement_text: bag of langstring_type(1000),
   statement_token: vocabulary_type,
)

vocabulary_type = record
(
   source: characterstring(iso-10646),
   value: characterstring(iso-10646),
)
```

Рис. 24. Определение компетенции

зованы *condition (условие)*, *action (действие)*, *standard (стандарт)*, *outcome (результат)*, *criteria (критерий)*.

Statement_ text представляет собой неструктурированное текстовое описание свойства компетенции, представленного данным утверждением. В качестве описания *statement_ text* может быть, например, текстовая строка «Задано множество целых чисел в диапазоне от 1 до 49».

Лексема *statement_ token* представляет собой словарную лексему, которая включает в себя *значение лексемы (value)* и *идентификатор схемы лексемы (source)*. Словарная лексема позволяет использовать зафиксированные словарные термины вместе (или вместо) со свободным текстовым описанием утверждения *statement_ text*.

Граф компетенций, описываемый стандартом IEEE SRCM, представляет собой ориентированный ациклический граф. Узлы графа компетенции могут ссылаться на различные формальные описания компетенций, структура которых была описана выше.

Граф компетенций представляет собой *нестрогую* иерархию узлов в том смысле, что узел графа компетенций может иметь более одного родительского узла. Если узел графа компетенций A имеет сыновей B и C, то из этого следует, что компетенция A состоит из суб-компетенций B и C, или, что компетенции B и C составляют A. На рис. 25 приведены четыре примера графа компетенций. Стрелка задает связь отец-сын.

Узел графа компетенций может ссылаться как на формальное определение компетенции, так и на другие графы компетенций. Указанная структура модели представлена на рис. 26.

Узлы графа компетенций могут иметь специальные правила, определяющие поведение системы при анализе графа компетенций. В правилах могут быть использованы следующие методы обработки компетенций:

- *метод all*,
- *метод any*,
- *метод fraction*,
- *метод units*,
- *метод mean*,
- *метод other*.

Метод all определяет правило, при котором для приобретения компетенции данного узла должны быть приобретены все компетенции, связанные с дочерними узлами. Этот метод используется по умолчанию.

Метод any определяет правило, при котором для приобретения компетенции данного узла должна быть приобретена хотя бы одна компетенция, связанная с одним из дочерних узлов.

Метод fraction определяет правило, при котором для приобретения компетенции данного узла должна быть приобретена определенная доля компетенций, ассоциированных с дочерними узлами. Значение доли является вещественным числом из отрезка [0, 1]. Значение 0 означает, что для приобретения компетенции не требуется приобретать дочерние компетенции, значение 1 эквивалентно методу all, значение 0.5 означает, что для приобретения компетенции требуется приобрести 50% дочерних компетенций.

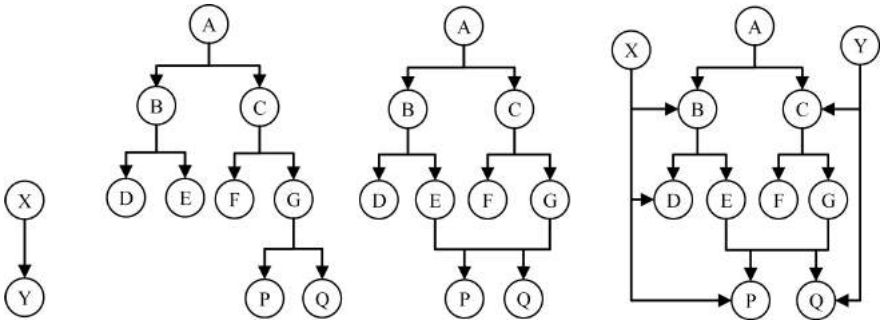

Рис. 25. Примеры графа компетенций с разным уровнем сложности

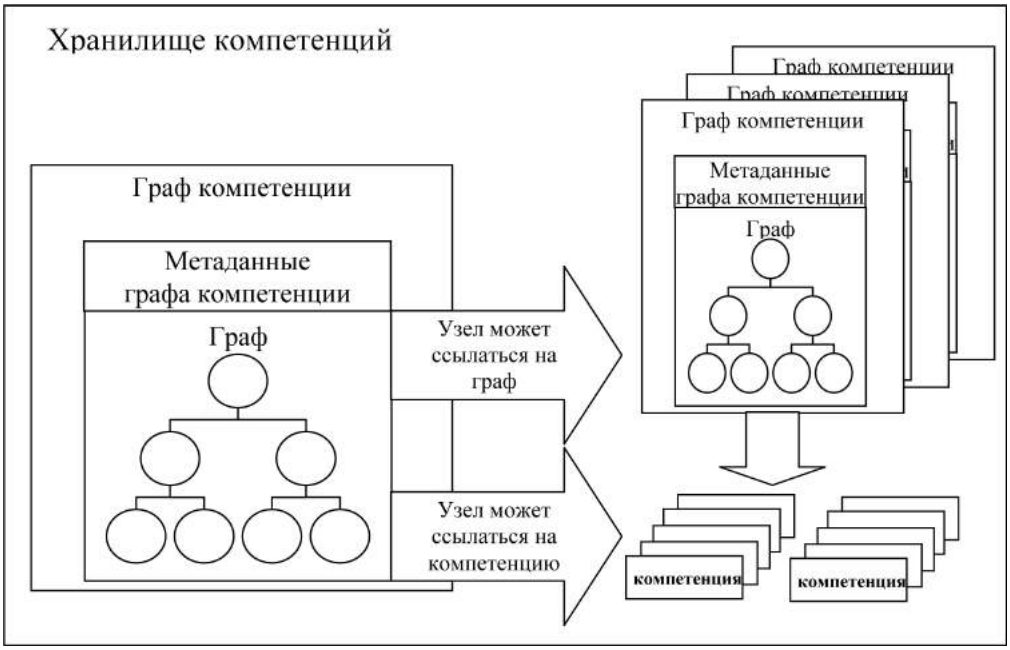

Рис. 26. Структура модели

Метод units определяет правило, при котором для приобретения компетенции данного узла должно быть приобретено определенное количество компетенций, ассоциированных с дочерними узлами. Количество является целочисленным значением. Значение 0 означает, что для приобретения компетенции не требуется приобретать дочерние компетенции, значение 1 эквивалентно *методу any*.

Метод mean определяет правило, при котором вычисляется мера приобретения компетенции, ассоциированной с данным узлом, как усредненное значение мер приобретения компетенций, связанных с дочерними узлами. Факт приобретения компетенции, ассоциированной с данным узлом, определяется путем сравнения вычисленной меры с заданным пороговым значением (proficiencyRequired).

Метод other определяет нестандартное правило, которое должно быть определено в профиле приложения.

Заключение

В работе предложен обзор основных международных стандартов на структуру и представление элементов образовательного контента в области электронного обучения. Базовым стандартом здесь является SCORM. Он обеспечивает возможность переноса элементов контента из одного электронного учебного курса в другой на физическом уровне.

Следует отметить, что до сих пор отсутствуют стандарты, определяющие принципы формирования дидактической структуры электронных учебных курсов. Это ограничивает возможность переноса методических наработок из одного электронного учебного курса в другой и препятствует получению максимального эффекта при внедрении электронного обучения в высшей школе. Авторами в настоящее время ведется работа по созданию высокоуровневой дидактической модели электронного учебного комплекса UniCST [24–26]. Модель предусматривает двухуровневую методическую базу знаний. Первый уровень предполагает создание комплекса электронных энциклопедий по различным областям знаний. Второй уровень предполагает создание электронных учебных курсов на основе существующих энциклопедий путем экспорта учебных блоков и организации их в иерархическую структуру, адекватно реализующих рабочие учебные программы в соответствии с образовательными стандартами направлений третьего поколения. Модель поддерживает структурирование учебного материала по способам представления образовательного контента. Учебный блок представляет собой набор компонент. Модель поддерживает стандартный набор таких компонент: теоретическое описание понятия; пример, иллюстрирующий те или иные отличительные черты понятия; упражнение для самостоятельного выполнения; тестовое задание; слайд презентации; библиографическая ссылка. Каждая компонента обладает своими дидактическими возможностями. Модель допускает возможность расширения набора стандартных дидактических способов представления учебного материала с учетом специфики каждой конкретной специальности или направления подготовки. Один и тот же блок учебного материала электронной энциклопедии может быть использован в разных электронных учебных курсах.

Литература

1. Tavangarian, D. Is e-Learning the Solution for Individual Learning? / D. Tavangarian, M. Leypold, K. Nōlting, M. Rōser // Electronic Journal of e-Learning. — 2004. — Vol. 2, No. 2. — URL: http://www.ejel.org/volume-2/vol2-issue2/v2-i2-art4-tavangarian.pdf (дата обращения: 02.10.2010).
2. Friedland, G. Architecting Multimedia Environments for Teaching. / Friedland G., Pauls K. — IEEE Computer Society, 2005. — Vol. 38, No. 6. — pp. 57–64.
3. Рут, С. Оправданно ли электронное обучение? / С. Рут // СУБД. — Открытые системы, 2010. — No. 03. — URL: http://www.osp.ru/os/2010/03/13001922/ (дата обращения: 02.09.2013).
4. Ли, К. и др. Новые Internet-технологии электронного обучения. // СУБД. — Открытые системы, 2009. — No. 07. — URL: http://www.osp.ru/os/2009/07/10456963/ (дата обращения: 02.09.2013).
5. IEEE 1484.11.1. Standard for Learning Technology — Data Model for Content Object Communication. 2004.
6. IEEE 1484.11.2. Standard for Learning Technology — ECMA Script Application Programming Interface for Content to Runtime Services Communication. 2003.
7. IEEE 1484.11.3. Standard for Learning Technology — Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication. 2005.
8. IEEE 1484.12.1. Draft Standard for Learning Object Metadata. 2002. URL: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf (дата обращения: 28.01.2013).
9. IEEE 1484.12.3. Standard for Learning Technology — Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata. 2005.
10. CRS003. Hierarchy of CBT Terms for AICC Publications.1992. URL: http://archive.aicc.org/docs/tech/crs003.rtf (дата обращения: 15.03.2011).
11. CMI001. CMI Guidelines for Interoperability AICC. 2004. URL: http://archive.aicc.org/docs/tech/cmi001v4.pdf (дата обращения: 15.03.2011).
12. IMS Content Packaging Information Model. Version 1.2. 2007. URL: http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_infov1p2pd2.html (дата обращения: 04.06.2011).
13. IMS Content Packaging XML Binding. Version 1.2. 2007. URL: http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_bindv1p2pd2.html (дата обращения: 04.06.2011).
14. IMS Simple Sequencing Best Practice and Implementation Guide, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bestv1p0.html (дата обращения: 10.06.2011).
15. IMS Simple Sequencing Information and Behavior Model, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_infov1p0.html (дата обращения: 10.06.2011).

16. IMS Simple Sequencing XML Binding, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bindv1p0.html (дата обращения: 10.06.2011).

17. SCORM 2004 4th Edition. Content Aggregation Model (CAM). Advanced Distributed Learning (ADL) Initiative. 2009.

18. SCORM 2004 4th Edition. Run-Time Environment (RTE). Advanced Distributed Learning (ADL) Initiative. 2009.

19. SCORM 2004 4th Edition. SCORM Users Guide for Programmers. Advanced Distributed Learning (ADL) Initiative. 2011.

20. SCORM 2004 4th Edition. Sequencing and Navigation (SN).Advanced Distributed Learning (ADL) Initiative. 2009.

21. Rifon L.A. Standardising competency definitions for engineering education. Proceedings of Global Engineering Education Conference (EDUCON), Amman, Jordan, 4–6 April 2010. IEEE, 2010. P. 52–58.

22. IEEE 1484.20.1. Standard for Learning Technology – Data Model for Reusable Competency Definitions. 2007.

23. IEEE Draft Standard on Simple Reusable Competency Map. 2006. URL: http://ieeeltsc.files.wordpress.com/2009/03/reusablecompetencymapproposal.pdf (дата обращения: 28.01.2013).

24. Силкина, Н.С. Модель образовательного стандарта третьего поколения на основе компетентностного подхода для систем электронного обучения / Н.С. Силкина, А.С. Евдокимова // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». — 2011. — № 37(254). Вып. 10. — С. 90–98.

25. Силкина, Н.С. Система UniCST — универсальная среда электронного обучения / Н.С. Силкина, Л.Б. Соколинский // Системы управления и информационные технологии. — 2010. — № 2. — С. 81–86.

26. Жигальская, Н.С. Моделирование дидактической структуры электронных учебных комплексов // Вестник Южно-Уральского государственного университета. Серия «Математическое моделирование и программирование». — 2008. — № 27(127). Вып. 2. — С. 4–9.

Силкина Надежда Сергеевна, ст. преподаватель кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), zhnadya@rambler.ru.

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

Поступила в редакцию 7 июля 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 5–35

DOI: 10.14529/cmse140401

E-LEARNING MODELS AND STANDARDS

N.S. Silkina, L.B. Sokolinsky

The paper is a review of models and standards used in modern e-learning systems. The general concept model of e-learning environment is considered. The review also regards these topics: the data model for interaction with learning objects; the content aggregation model which describes the structure of learning objects, the ways of searching, packing and transmitting them between different learning systems; the run-time environment model, which defines an API for learning objects management; the sequencing model for learning content; the competency model, which is used to specify knowledge, skills and abilities in e-learning systems. The paper also reviews SCORM, which integrates a system of e-learning models.

Keywords: e-learning, metadata model, data model, content aggregation model, run-time environment, simple sequencing information and behavior model, competency model, SCORM.

References

1. Tavangarian D., Leypold M., Nōlting K., Rōser M. Is e-Learning the Solution for Individual Learning? Electronic Journal of e-Learning. 2004. Vol. 2, No. 2. URL: http://www.ejel.org/volume-2/vol2-issue2/v2-i2-art4-tavangarian.pdf.
2. Friedland G., Pauls K. Architecting Multimedia Environments for Teaching. IEEE Computer Society. 2005. Vol. 38, No. 6. pp. 57–64. DOI: 10.1109/MC.2005.181.
3. Ruth S. Is E-Learning Really Working? The Trillion-Dollar Question. IEEE Internet Computing. 2010. Vol. 14,, No. 2. pp. 78–82. DOI: 10.1109/MIC.2010.46.
4. Li Q., Lau R., Leung R., Li F., Lee V., Wah B., Ashman H. Emerging Internet Technologies for E-Learning. IEEE Internet Computing. 2009. Vol. 13, No. 4. pp. 11–17. DOI: 10.1109/MIC.2009.83.
5. IEEE 1484.11.1. Standard for Learning Technology — Data Model for Content Object Communication. 2004.
6. IEEE 1484.11.2. Standard for Learning Technology — ECMA Script Application Programming Interface for Content to Runtime Services Communication. 2003.
7. IEEE 1484.11.3. Standard for Learning Technology — Extensible Markup Language (XML) Schema Binding for Data Model for Content Object Communication. 2005.
8. IEEE 1484.12.1. Draft Standard for Learning Object Metadata. 2002. URL: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf.
9. IEEE 1484.12.3. Standard for Learning Technology — Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata. 2005.
10. CRS003. Hierarchy of CBT Terms for AICC Publications.1992. URL: http://archive.aicc.org/docs/tech/crs003.rtf (дата обращения: 15.03.2011).
11. CMI001. CMI Guidelines for Interoperability AICC. 2004. URL: http://archive.aicc.org/docs/tech/cmi001v4.pdf (дата обращения: 15.03.2011).

12. IMS Content Packaging Information Model. Version 1.2. 2007. URL: http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_infov1p2pd2.html (дата обращения: 04.06.2011).
13. IMS Content Packaging XML Binding. Version 1.2. 2007. URL: http://www.imsglobal.org/content/packaging/cpv1p2pd2/imscp_bindv1p2pd2.html (дата обращения: 04.06.2011).
14. IMS Simple Sequencing Best Practice and Implementation Guide, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bestv1p0.html (дата обращения: 10.06.2011).
15. IMS Simple Sequencing Information and Behavior Model, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_infov1p0.html (дата обращения: 10.06.2011).
16. IMS Simple Sequencing XML Binding, IMS Global Learning Consortium. 2003. URL: http://www.imsglobal.org/simplesequencing/ssv1p0/imsss_bindv1p0.html (дата обращения: 10.06.2011).
17. SCORM 2004 4th Edition. Content Aggregation Model (CAM). Advanced Distributed Learning (ADL) Initiative. 2009.
18. SCORM 2004 4th Edition. Run-Time Environment (RTE). Advanced Distributed Learning (ADL) Initiative. 2009.
19. SCORM 2004 4th Edition. SCORM Users Guide for Programmers. Advanced Distributed Learning (ADL) Initiative. 2011.
20. SCORM 2004 4th Edition. Sequencing and Navigation (SN).Advanced Distributed Learning (ADL) Initiative. 2009.
21. Rifon L.A. Standardising competency definitions for engineering education. Proceedings of Global Engineering Education Conference (EDUCON), Amman, Jordan, 4–6 April 2010. IEEE, 2010. P. 52–58.
22. IEEE 1484.20.1. Standard for Learning Technology – Data Model for Reusable Competency Definitions. 2007.
23. IEEE Draft Standard on Simple Reusable Competency Map. 2006. URL: http://ieeeltsc.files.wordpress.com/2009/03/reusablecompetencymapproposal.pdf (дата обращения: 28.01.2013).
24. Silkina N.S., Evdokimova A.S. Model' obrazovatel'nogo standarta tret'ego pokolenija na osnove kompetentnostnogo podhoda dlja sistem jelektronnogo obuchenija [A Model of the Third Generation Educational Standard on the Basis of Competency Approach for E-learning Systems]. Bulletin of South Ural State University, "Mathematical Modelling, Programming & Computer Software" Series. 2011. Vol. 37(254). No. 10. pp. 90–98.
25. Silkina N.S., Sokolinsky L.B. Sistema UniCST — universal'naja sreda jelektronnogo obuchenija [UniCST System — a Universal E-learning Environment]. Sistemy upravlenija i informacionnye tehnologii [Control Systems and Information Technologies]. 2010. No. 2. pp. 81–86.
26. Zhigalskaya N.S. Modelirovanie didakticheskoj struktury jelektronnyh uchebnyh kompleksov [Didactic Structure Modeling of E-learning Systems]. Bulletin of South Ural State University, "Mathematical Modelling, Programming & Computer Software" Series. 2008. Vol. 27(127). No. 2. pp. 4–9.

Received July 7, 2014.

УДК 004.65, 004.622 DOI: 10.14529/cmse140402

АЛГОРИТМ РЕПРЕЗЕНТАТИВНОГО СЭМПЛИНГА ДЛЯ СИСТЕМ БАЗ ДАННЫХ НА ОСНОВЕ ФРАГМЕНТНОГО ПАРАЛЛЕЛИЗМА

Д.Д. Янцен, М.Л. Цымблер, В.Ю. Гудков

Сэмплинг является популярным подходом к обработке сверхбольших баз данных в широком спектре приложений, связанных с интеллектуальным анализом данных, построением гистограмм, приблизительное исполнение запросов и др. Использование сэмпла вместо оригинальной базы данных может уменьшить точность результатов, но компенсируется сокращением времени выполнения обработки. Репрезентативный сэмплинг позволяет сохранить в сэмпле определенные характеристики базы данных. Однако существующие алгоритмы репрезентативного сэмплинга не могут быть применены для параллельных систем баз данных, поскольку не учитывают характеристики данных, распределяемых по вычислительным узлам кластерной системы. В данной статье предлагается алгоритм репрезентативного сэмплинга для параллельных реляционных систем баз данных на основе фрагментного параллелизма. Приведены результаты вычислительных экспериментов над предложенным алгоритмом, показавшие адекватное сохранение репрезентативности свойств базы данных, распределенной по узлам кластерной системы.

Ключевые слова: реляционные базы данных, параллельные системы баз данных, репрезентативный сэмплинг.

Введение

В настоящее время *сэмплинг* применяется в широком спектре приложений, связанных с обработкой сверхбольших баз данных: интеллектуальный анализ данных [23, 25], построение гистограмм [18], генерация баз данных для тестирования программного обеспечения [26], приблизительное исполнение запросов [5] и др. Сэмпл представляет собой часть оригинальной базы данных, имеющую меньший размер, что позволяет сократить время обработки базы данных за счет возможной потери точности результатов обработки. *Случайный сэмплинг* предполагает отбор кортежей базы данных без учета значений их атрибутов, что снижает точность и повторяемость результатов обработки. При *репрезентативном сэмплинге* отбор кортежей осуществляется с сохранением важных особенностей оригинальной базы данных: относительное количество кортежей, связанных посредством внешних ключей, относительное количество значений атрибута и др. [10].

На сегодня научное сообщество признает *параллельные системы баз данных* [3] как практически единственное эффективное средство для организации обработки сверхбольших баз данных. Базисной концепцией параллельных систем баз данных является *фрагментный параллелизм*, предполагающий разбиение отношений базы данных на горизонтальные фрагменты, которые могут обрабатываться независимо на разных узлах кластерной вычислительной системы. Однако существующие методы репрезентативного сэмплинга не приспособлены для систем баз данных на основе фрагментного параллелизма, поскольку не учитывают особенности распределения фрагментов отношений базы данных по вычислительным узлам кластерной системы.

В данной работе предлагается алгоритм репрезентативного сэмплинга реляционной базы данных для параллельных систем баз данных. Статья организована следующим образом. В разделе 1 представлен обзор существующих методов сэмплинга реляционных баз данных. Раздел 2 содержит описание предлагаемого алгоритма. В разделе 3 рассмотрены результаты вычислительных экспериментов, в которых исследуются свойства предложенного алгоритма. В заключении суммируются полученные результаты и рассматриваются направления дальнейших исследований в данной области.

1. Обзор работ в области сэмплинга данных

В настоящее время существует большое количество подходов к сэмплингу баз данных, которые, в основном, ориентированы на конкретные приложения: тестирование программного обеспечения, интеллектуальный анализ данных, нахождение приблизительного результата исполнения запросов и др. [10].

Случайный сэмплинг является одним из первых предложенных и хорошо разработанных подходов к решению проблемы обработки сверхбольших баз данных [21]. В работе [7] представлен алгоритм случайного сэмплинга, позволяющий сохранить в сэмпле ограничения целостности оригинальной базы данных: ссылочная целостность по внешним ключам, функциональные зависимости, ограничения целостности доменов.

В настоящее время сэмплинг поддерживается в ряде СУБД корпоративного класса. В СУБД Oracle поддерживается выборка данных из случайного сэмпла указанной таблицы путем добавления в запрос SELECT специального ключевого слова SAMPLE [22]. СУБД IBM DB2 предоставляет два оператора, обеспечивающих сэмплинг, — RAND и TABLESAMPLE [15]. Функция RAND(), указанная в части WHERE запроса, позволяет задает долю кортежей оригинальной таблицы, случайно попадающих в сэмпл. Оператор TABLESAMPLE позволяет отбирать в кортежи в сэмпл одним из двух способов: на основе алгоритма, предложенного в работе [13], или случайным отбором физических страниц диска, на которых размещены кортежи.

В области интеллектуального анализа данных сэмплинг является одним из наиболее часто используемых методов для достижения баланса между вычислительной сложностью аналитической обработки большого массива данных и точностью получаемых результатов. Однако соответствующие алгоритмы сэмплинга, как правило, ориентированы на конкретные алгоритмы интеллектуального анализа, которые будут использовать сэмпл в качестве входных данных, и могут применяться только для реляционных баз данных, состоящих из единственного отношения [14, 19]. В работе [19] предложен статический метод сэмплинга, который использует распределение данных сэмпла как критерий принятия решения, отражает ли сэмпл оригинальное множество данных. Данный подход, однако, ограничен применением для случая базы данных, состоящей из одного отношения. В работе [27] представлен алгоритм сэмплинга для реляционных баз данных, направленный на улучшение масштабируемости и точности методов классификации для мульти-реляционных данных.

Большое количество работ посвящено использованию случайного сэмплинга для быстрого нахождения приблизительного результата запроса [6, 17] и размера результирующего отношения [12]. В работе [17] предложен метод Linked Bernoulli Synopses, основанный на алгоритме Join Synopses (JS) [5] для вычисления приблизительного результата выполнения запроса, предполагающего соединение

большого количества отношений. Оба подхода требуют обработки каждого кортежа базы данных. В случае JS каждый кортеж базы данных попадает в сэмпл с вероятностью, равной коэффициенту сэмплинга (отношению количества кортежей в сэмпле к кортежей в оригинальной базе данных). После вставки кортежей в сэмпл JS гарантирует сохранение ссылочной целостности в сэмпле, дополнительно выполняя последовательный перебор всех отношений сэмпла, начиная с некоторого стартового отношения, и добавляя в каждое отношение пропущенные ссылаемые кортежи. Алгоритм LBS предполагает однократный запуск над всей базой данных. Для каждого кортежа в каждом отношении оригинальной базы данных вычисляется вероятность вставки данного кортежа в сэмпл, которая зависит от вероятностей вставки в сэмпл кортежей, ссылающихся на этот кортеж.

Алгоритм VFDS (Very Fast Database Sampling) [9] является одним из наиболее быстрых алгоритмов случайного сэмплинга баз данных. VFDS требует одного прохода над всей оригинальной базой данных и не выполняет обработку каждого ее кортежа в отдельности. На первом шаге случайным образом отбираются кортежи стартового отношения и вставляются в соответствующее отношение сэмпла. После этого алгоритм обеспечивает целостность данных в сэмпле посредством сэмплинга ссылающихся и ссылаемых кортежей стартового отношения. Процесс рекурсивно продолжается для ссылающихся и ссылаемых кортежей, попавших в сэмпл на предыдущем шаге. Сэмплинг завершается, как только обработаны все отношения оригинальной базы данных.

Репрезентативность является важным свойством сэмплинга, которое обеспечивает меньшие время и вычислительную сложность обработки сэмпла по сравнению с обработкой оригинальной базы данных, одновременно сохраняя похожесть результатов обработки. Репрезентативный сэмпл реляционной базы данных определяется [10] как сэмпл, отбор в который осуществляется таким образом, чтобы сохранить то же, что и в оригинальной базе данных, распределение значений определенных атрибутов. Такими атрибутами, прежде всего, являются внешние ключи, поскольку они реализуют логические связи между кортежами различных отношений.

Алгоритм CoDS (Chains of Dependencies-based sampling) [10] является на сегодня, по-видимому, самым быстрым алгоритмом репрезентативного сэмплинга. Ключевой идеей алгоритма является определение кортежей, которые должны быть отобраны из стартового отношения. После того, как эти кортежи будут вставлены в сэмпл базы данных, выполняется вставка в сэмпл ссылающихся и ссылаемых кортежей из оставшихся отношений оригинальной базы данных. Подобно алгоритму VFDS, процесс является рекурсивным и завершается, когда просмотрены все отношения. Отличие заключается в том, что кортежи стартового отношения отбираются в сэмпл таким образом, чтобы в итоге обеспечить необходимую репрезентативность.

Разработанный нами алгоритм репрезентативного сэмплинга для параллельных систем баз данных, представленный в разделе 2, является модернизированной версией алгоритма CoDS. В силу этого далее мы приводим краткое описание данного алгоритма. CoDS состоит из четырех следующих этапов: выбор стартового отношения, нахождение цепочек зависимостей и построение соответствующих диаграмм рассеивания, анализ диаграмм рассеивания и финальный отбор кортежей в сэмпл.

На этапе *выбора стартового отношения* в качестве стартового берется любое отношение базы данных, не имеющее внешних ключей. Если таких отношений несколько, в качестве стартового выбирается то из них, которое имеет максимальную мощность.

Цепочка зависимостей (chain of dependencies) представляет собой последовательность, в которой перечислены два или более отношений оригинальной базы данных. В этой последовательности в двух любых соседних членах одно из них ссылается на другое посредством внешнего ключа. На втором этапе осуществляется поиск всех цепочек зависимостей, начинающихся со стартового отношения. Затем для каждой найденной цепочки осуществляется построение диаграммы рассеивания. *Диаграмма рассеивания (scatter plot)* представляет собой гистограмму, по оси абсцисс которой откладывается количество кортежей стартового отношения, а по оси ординат — количество кортежей из конечного отношения из цепочки зависимостей, для которой строится диаграмма. Точка на этой гистограмме означает, что x кортежей из стартового отношения связаны с y кортежей из конечного отношения цепочки. Кортежи стартового отношения, связанные с данной точкой на диаграмме рассеивания, называются *точкой данных (data point)*.

На этапе *анализа диаграмм рассеивания* осуществляется отбор кортежей стартового отношения, которые будут включены в сэмпл. Основной задачей здесь является уменьшение размеров каждой точки данных в соответствии с заданным коэффициентом сэмплинга. Для этого авторами алгоритма предложена *функция влияния (impact factor, IF)*, которая для заданного кортежа вычисляет степень нарушения репрезентативности сэмпла в случае попадания данного кортежа в сэмпл. Чем меньше значение функции влияния, тем больше вероятность включения в сэмпл данного кортежа.

На этапе *финального заполнения сэмпла* в отношения базы данных добавляются кортежи, которые связаны с кортежами, отобранными в стартовое отношение посредством внешних ключей.

В следующем разделе мы покажем, каким образом можно модернизировать алгоритм CoDS для параллельных реляционных систем баз данных на основе фрагментного параллелизма.

2. Алгоритм pCoDS репрезентативного сэмплинга параллельных систем баз данных

Рассмотренные в предыдущем разделе методы сэмплинга не могут быть применены для параллельных систем баз данных на основе фрагментного параллелизма, поскольку не учитывают особенности распределения фрагментов отношений базы данных по вычислительным узлам кластерной системы. Далее мы рассмотрим разработанную нами модернизированную версию алгоритма CoDS, названную нами *pCoDS* [4]. В pCoDS при построении сэмпла делается попытка сохранить в нем следующие важные особенности оригинальной базы данных: соотношение размеров фрагментов отношений и соотношение «своих» и «чужих» кортежей в отношениях. «Своими» называют такие кортежи, которые при выполнении запроса должны быть обработаны на текущем вычислительном узле кластера, «чужими» — те, что должны быть переданы для обработки на другой вычислительный узел.

Сохранение относительных размеров фрагментов отношений направлено на обеспечение репрезентативности *перекоса по данным*, когда значение некоторые значения для

определенного атрибута встречаются значительно чаще, чем остальные. Перекос по данным может привести к дисбалансу загрузки узлов кластера и катастрофически ограничить ускорение параллельной СУБД [20]. Поскольку фрагментация часто осуществляется на основе значений первичного ключа отношения, сохранение относительных размеров фрагментов отношений обеспечивает репрезентативность возможного дисбаланса загрузки вычислительных узлов кластера при выполнении запроса. Сохранение соотношения «своих» и «чужих» кортежей обеспечивает репрезентативность нагрузки, связанной с пересылкой данных между вычислительными узлами кластера при выполнении запроса.

В алгоритме pCoDS нами модифицируются следующие шаги исходного алгоритма: выбор стартового отношения, создание диаграмм рассеивания и анализ диаграмм рассеивания.

На этапе *выбора стартового отношения* в алгоритме CoDS выбирается отношение, не содержащее внешних ключей. Предложенная нами модификация алгоритма подразумевает, что в качестве стартового выбирается отношение, которое наиболее часто фигурирует в запросах к базе данных. Данная модификация направлена на обеспечение репрезентативности соотношения «своих» и «чужих» кортежей для запросов, содержащих в соединениях стартовое отношение.

На этапе *создания диаграмм рассеивания* мы вносим изменения в принцип построения диаграмм рассеивания. Для каждой цепочки зависимостей в pCoDS строится *две диаграммы рассеивания*: для «своих» кортежей (обработка которых осуществляется на текущем узле) и для «чужих» кортежей (обработка которых требует пересылки на другой узел кластера) стартового отношения. В силу того, что размер всех точек данных из каждой диаграммы рассеивания уменьшается пропорционально коэффициенту сэмплинга, данная модификация позволяет сохранить соотношение «своих» и «чужих» кортежей для отношения, выбранного стартовым в алгоритме.

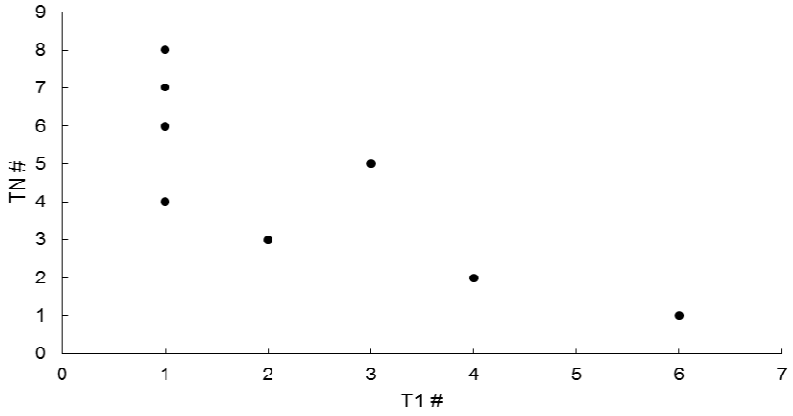

Рис. 1. Диаграмма рассеивания в CoDS

Пример диаграммы рассеивания для цепочки (T_1, T_2, T_3, ..., T_N) в алгоритме CoDS приведен на рис. 1. Диаграммы рассеивания для цепочки (T_1, T_2, T_3, ..., T_N) в алгоритме pCoDS приведены на рис. 2, здесь φ означает функцию фрагментации отношения.

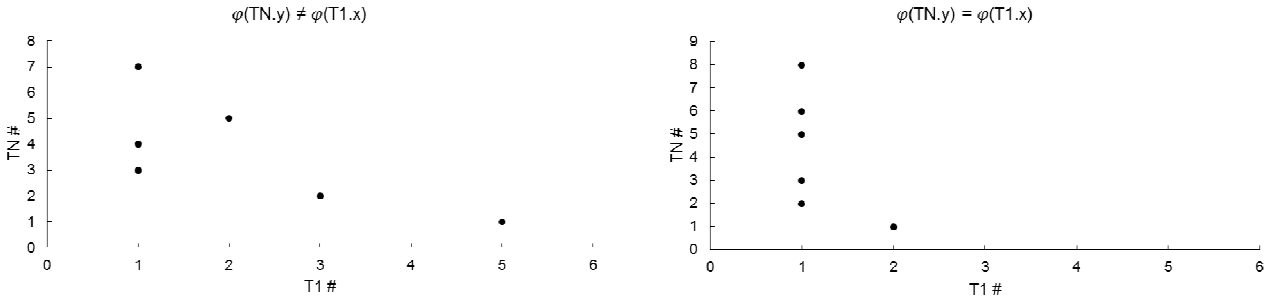

Рис. 2. Диаграммы рассеивания в алгоритме pCoDS

Запросы, используемые для создания диаграмм рассеивания в алгоритме pCoDS, приведены на рис. 3.

```
SELECT T1.ID, COUNT(DISTINCT TN.ID) AS Y
FROM T1 JOIN T2 JOIN T3 JOIN … JOIN TN
WHERE φ(T1.ID) == φ(TN.ID)
GROUP BY T1.ID

SELECT T1.ID, COUNT(DISTINCT TN.ID) AS Y
FROM T1 JOIN … JOIN TN
WHERE φ(T1.ID) <> φ(TN.ID)
GROUP BY T1.ID
```

Рис. 3. Запросы для создания диаграмм рассеивания в pCoDS

На этапе *анализа диаграмм рассеивания* мы изменяем функцию влияния, добавляя в нее дополнительное слагаемое с целью обеспечить репрезентативность относительных размеров фрагментов базы данных, и функция влияния принимает вид (1). В силу того, что функция влияния рассчитывается для каждого кортежа на основе кортежей, уже отобранных в сэмпл, предложенная модификация позволяет сохранить данную характеристику. С помощью коэффициента k регулируется степень важности сохранения данной характеристики.

$$IF(T^*.t) = IF_{CoDS}(T^*.t) + k * \frac{\left\|TS^*_{\varphi(t)}\right\| / \|TS^*\|}{\left\|T^*_{\varphi(t)}\right\| / \|T^*\|} \quad (1)$$

где

TS^* — стартовое отношение в сэмпле базы данных,

T^* — стартовое отношение в исходной базе данных,

φ — функция фрагментации стартового отношения,

t — обрабатываемый кортеж,

$k \in [0, 1]$ — коэффициент, определяющий степень важности сохранения относительных размеров фрагментов,

$IF_{CoDS}(T^*.t)$ — функция влияния исходного алгоритма CoDS, вычисляемая по формуле (2).

$$IF_{CoDS}(T^*.t) = \sum_{dp' \in RDP(dp)} \frac{\|dp' \cap TS^*\|}{\alpha * \|dp'\|} \quad (2)$$

где

dp — точка данных на диаграмме рассеивания,

RDP(dp) — множество точек данных, содержащих одинаковые кортежи (связанные точки данных, related data points),

a — коэффициент сэмплинга.

3. Вычислительные эксперименты

Предложенный нами алгоритм был реализован для параллельной СУБД PargreSQL [2, 24]. Для исследования свойств реализованного алгоритма нами проведена серия вычислительных экспериментов, в которых использовалась финансовая база данных [16], предлагавшаяся участникам конкурса PKDD'99 Challenge Discovery, и использованная в экспериментах разработчиками алгоритма CoDS. В рамках экспериментов исследовались следующие характеристики предложенного алгоритма: репрезентативность сэмпла, отклонение размера результирующего сэмпла, репрезентативность загрузки узлов кластера и репрезентативность нагрузки по пересылке кортежей между узлами при обработке запросов.

Репрезентативность базы данных. На рис. 4 приведены результаты экспериментов по измерению репрезентативности сэмпла. Можно видеть, что при увеличении коэффициента важности сохранения относительных размеров фрагментов, репрезентативность сэмпла ухудшается.

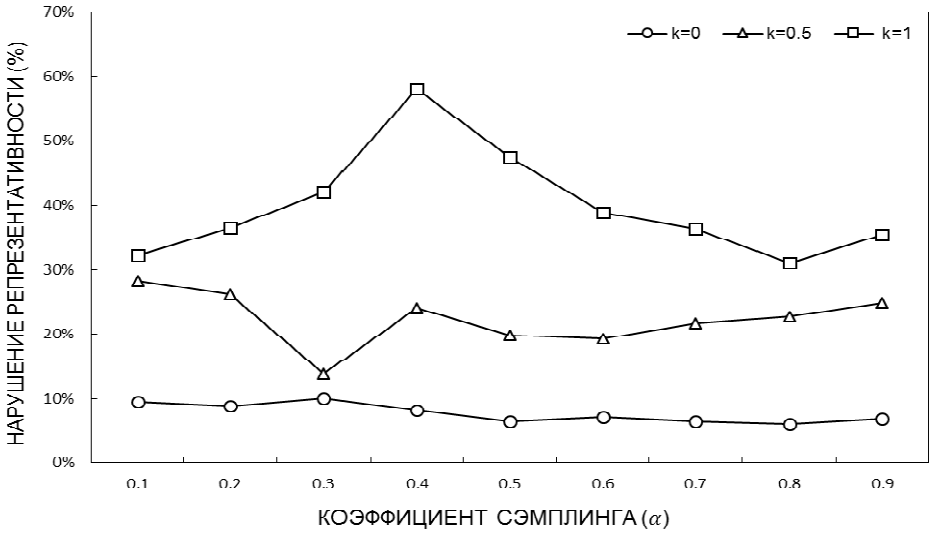

Рис. 4. Репрезентативность сэмпла

Для измерений нами использовалась мера (3), предложенная авторами алгоритма CoDS. Результирующее значение, равное нулю, будет означать идеальную репрезентативность сэмпла.

$$\delta(S(T)) = \frac{1}{\|S(T)\|}\sum_{t\in T}\left(\frac{1}{\|RT(t)\|}\sum_{t'\in RT(t)}\left(\frac{1}{\left\|sp_t^{t'}\right\|}\sum_{dp_t^{t'}\in sp_t^{t'}}\min\left(\frac{|C-\lfloor E\rfloor|}{\lfloor E\rfloor},\frac{|C-\lceil E\rceil|}{\lceil E\rceil}\right)\right)\right) \tag{3}$$

где

t', *t* — отношения базы данных,

$\left\|sp_t^{t'}\right\|$ — количество точек данных на диаграмме рассеивания между *t'* и *t*,

$dp_t^{t'}$ — точка данных из диаграммы рассеивания $sp_t^{t'}$,

$S(T)$ — сэмпл базы данных,

$C = \| dp_t^{t'} \cap S(T) \|$ — количество кортежей в точке данных в сэмпле,

$E = a * \| dp_t^{t'} \|$ — ожидаемое количество кортеже в точке данных в сэмпле,

$RT(t)$ — отношения, связанные с t.

Отклонение размера сэмпла от заданного. На рис. 5 приведены результаты экспериментов по измерению отклонения размера сэмпла от заданного. Можно видеть, что увеличение коэффициента сохранения относительных размеров фрагмент не влияет на получаемый размер сэмпла. При этом сэмпл получается меньше, чем задаваемое значение в среднем на 18 %, что связано с особенностями алгоритма CoDS и является ожидаемым результатом.

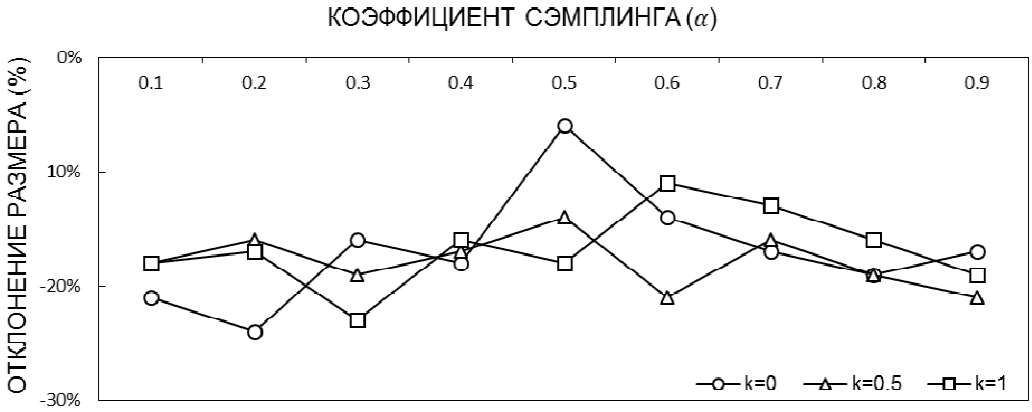

Рис. 5. Отклонение размера результирующего сэмпла от заданного

Для измерения соответствия размера сэмпла заданному использовалась мера (4), предложенная авторами алгоритма CoDS. Результирующее значение, равное нулю, будет означать идеальное соответствие размера сэмпла заданному.

$$\delta_{\|T\|}(S(T)) = \frac{\|S(T)\| - \alpha * \|O(T)\|}{\alpha * \|O(T)\|} \tag{4}$$

Репрезентативность загрузки узлов кластера. Для измерения сохранения относительных размеров фрагментов нами предлагается мера (5). Результирующее значение, равное нулю, будет означать идеальную репрезентативность загрузки узлов базы данных.

$$\delta_{\varphi}(S(T)) = \frac{1}{n}\sum_{k=1}^{n}\left|\frac{\|S(T)_k\|}{\|S(T)\|} - \frac{\|O(T)_k\|}{\|O(T)\|}\right| \tag{5}$$

где

n — количество фрагментов,

$S(T)$ — сэмпл базы данных,

$O(T)$ — исходная база данных.

На рис. 6 приведены результаты экспериментов по измерению репрезентативности загрузки узлов кластера. При увеличении коэффициента важности сохранения относительных размеров фрагментов, степень сохранения относительных размеров фрагментов (5)

улучшается. При сравнении результатов при k=0 и k=1 среднее значение меры уменьшается почти вдвое с 0,236 до 0,124. Это позволяет заключить, что модификация функции влияния до вида (1) действительно улучшает репрезентативность дисбаланса загрузки узлов кластера.

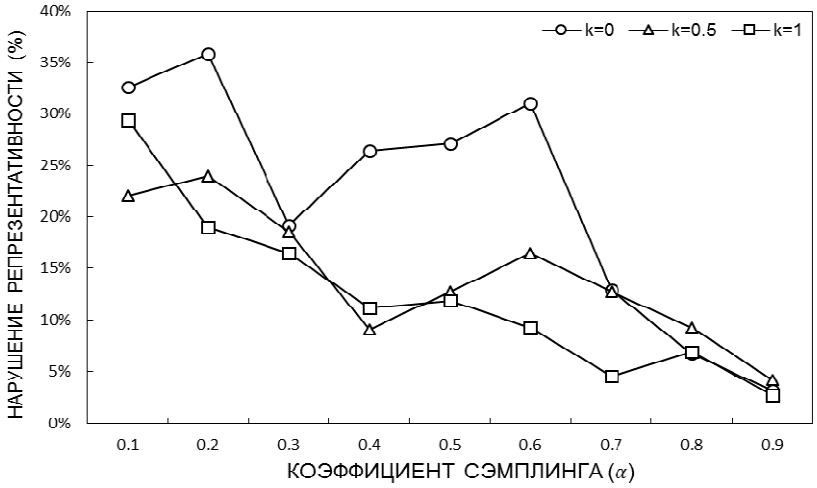

Рис. 6. Репрезентативность загрузки узлов кластера

Репрезентативность нагрузки по пересылке данных между узлами. Для измерения сохранения соотношения «своих» и «чужих» кортежей нами предлагается мера (6). Измерение проводилось на двух типах запросов: содержащих и не содержащих стартовое отношение в соединении.

$$\delta_{io} = \left| \frac{native_{S(T)}}{alien_{S(T)}} - \frac{native_{O(T)}}{alien_{O(T)}} \right| \tag{6}$$

где

$native_X$ — количество «своих» кортежей при запросе к базе данных X,

$alien_X$ — количество «чужих» кортежей при запросе к базе данных X.

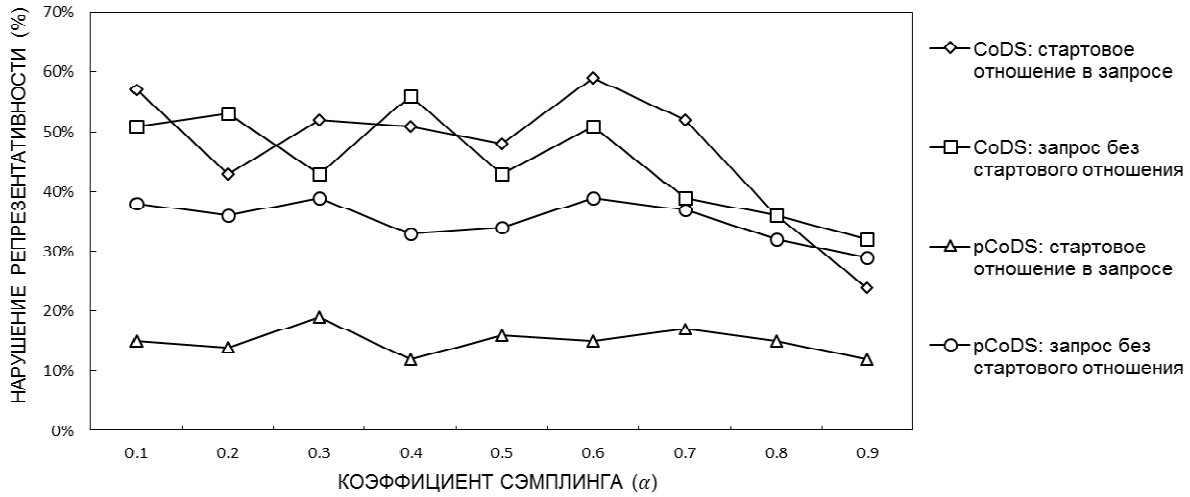

Рис. 7. Репрезентативность нагрузки по пересылке данных между узлами кластера

Результаты экспериментов по измерению репрезентативности возможной нагрузки по пересылке данных между узлами приведены на рис. 7. Можно видеть, что модификация

шага алгоритма, связанного с построением диаграмм рассеивания, позволяет сильно улучшить репрезентативность данной характеристики для запросов, содержащих в соединениях стартовое отношение. При этом для запросов, не содержащих стартовое отношение, степень репрезентативности также улучшается. Результаты показывают, что для запросов, содержащих стартовое отношение в соединениях, среднее значение меры уменьшается с 0,43 до 0,15, что является отличным результатом. При этом для запросов, не содержащих стартового отношения, среднее значение уменьшается с 0,44 до 0,36, что также является хорошим результатом.

Таким образом, результаты проведенных экспериментов свидетельствуют о том, что предложенный алгоритм сэмплинга pCoDS адекватно сохраняет важные характеристики параллельной системы базы данных и обеспечивает приемлемую репрезентативность данных.

Заключение

В статье рассмотрена разработка алгоритма репрезентативного сэмплинга для параллельных систем баз данных на основе фрагментного параллелизма. Предложена модификация существующего алгоритма CoDS для последовательных систем баз данных, который является одним из наиболее быстрых алгоритмов сэмплинга и показывает при этом приемлемую репрезентативность. Модифицированный алгоритм позволяет сохранять при сэмплинге следующие важные для параллельных СУБД характеристики распределения базы данных по вычислительным узлам кластерной системы: относительные размеры фрагментов отношений и относительное количество «своих» и «чужих» кортежей в каждом фрагменте отношения. Сохранение относительных размеров фрагментов обеспечивает репрезентативность возможного дисбаланса загрузки вычислительных узлов кластера при обработке запроса. Сохранение относительного количества «своих» и «чужих» кортежей (т.е. кортежей, обрабатываемых соответственно на текущем узле или пересылаемых на другой узел кластера) в фрагментах обеспечивает репрезентативность нагрузки, связанной с пересылкой данных между узлами кластера при выполнении запросов. Разработанный алгоритм реализован для параллельной СУБД PargreSQL. Представлены результаты вычислительных экспериментов на реальных данных, подтверждающие приемлемую репрезентативность предложенного алгоритма.

В качестве возможных направлений дальнейших исследований интересными представляются следующие задачи:

1) встраивание предложенного алгоритма сэмплинга в ядро параллельной СУБД PargreSQL для расширения синтаксиса запросов SELECT;

2) разработка меры репрезентативности базы данных, фрагментированной по узлам кластерной системы, и ее сэмпла, которая учитывала бы как статистические характеристики базы данных, так и рассмотренные в данной статье характеристики, важные для параллельной системы баз данных;

3) адаптация предложенного алгоритма сэмплинга для параллельных систем баз данных, поддерживающих частичную репликацию данных [1].

Исследование выполнено при финансовой поддержке Российского фонда фундаментальных исследований в рамках научного проекта № 12-07-00443-а.

Литература

1. 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.
2. 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.
3. Соколинский, Л.Б. Обзор архитектур параллельных систем баз данных. / Л.Б. Соколинский // Программирование. — 2004. — № 6. — С. 49–63.
4. Янцен, Д.Д. Алгоритм репрезентативного сэмплинга для параллельных систем баз данных. / Д.Д. Янцен, М.Л. Цымблер // Параллельные вычислительные технологии (ПаВТ'2014): труды международной научной конференции (1–3 апреля 2014 г., г. Ростов-на-Дону). — Челябинск: Издательский центр ЮУрГУ, 2014. — С. 381.
5. Acharya, S. Join Synopses for Approximate Query Answering. / S. Acharya, P.B. Gibbons, V. Poosala, S. Ramaswamy // SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1–3, 1999, Philadelphia, Pennsylvania, USA. — ACM, 1999. — P. 275–286.
6. Agarwal, S. Blink and It's Done: Interactive Queries on Very Large Data. / S. Agarwal, A. Panda, B. Mozafari, A.P. Iyer, S. Madden, I. Stoica // Proceedings of the VLDB Endowment. — 2011. — Vol. 5, No. 1. — P. 1902–1905.
7. Bisbal, J. A Formal Framework for Database Sampling. / J. Bisbal, J. Grimson, D.A. Bell // Information & Software Technology. — 2005. — Vol. 47, No. 1. — P. 819–828.
8. Buda, T.S. Generation of Test Databases using Sampling Methods. / T.S. Buda // International Symposium on Software Testing and Analysis, ISSTA'13, Lugano, Switzerland, July 15–20, 2013. — ACM, 2013. — P. 366–369.
9. Buda, T.S. VFDS: Very Fast Database Sampling System. / T.S. Buda, T. Cerqueus, M. Kristiansen, J. Murphy // Proceedings of the IEEE 14th International Conference on Information Reuse & Integration, IRI 2013, San Francisco, CA, USA, August 14–16, 2013. — IEEE, 2013. — P. 153–160.
10. Buda, T.S. CoDS: A Representative Sampling Method for Relational Databases. / T.S. Buda, T. Cerqueus, J. Murphy, M. Kristiansen // Database and Expert Systems Applications – 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26–29, 2013. Proceedings, Part I. — Springer, 2013. Lecture Notes in Computer Science. — Vol. 8055. — P. 342–356.
11. Chakaravarthy, V.T. Analysis of Sampling Techniques for Association Rule Mining. / V.T. Chakaravarthy, V. Pandit, Y. Sabharwal // Database Theory – ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23–25, 2009, Proceedings. — ACM, 2009. — P. 276–283.
12. Chaudhuri, S. Effective Use of Block–Level Sampling in Statistics Estimation. / S. Chaudhuri, G. Das, U. Srivastava // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13–18, 2004. — ACM, 2004. — P. 287–298.

13. Gemulla, R. Linked Bernoulli Synopses: Sampling along Foreign Keys. / R. Gemulla, P. Rösch, W. Lehner // Scientific and Statistical Database Management, 20th International Conference, SSDBM 2008, Hong Kong, China, July 9–11, 2008, Proceedings. — Springer, 2008. Lecture Notes in Computer Science. — P. 6–23.
14. Goethals, B. Mining Interesting Sets and Rules in Relational Databases. / B. Goethals, W.L. Page, M. Mampaey // Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, March 22–26, 2010. — ACM, 2010. — P. 997–1001.
15. Gryz, J. Query Sampling in DB2 Universal Database. / J. Gryz, J. Guo, L. Liu, C. Zuzarte // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13–18, 2004. — ACM, 2004. — P. 839–843.
16. Guide to the Financial Data Set of the PKDD'99 Discovery Challenge. — URL: http://lisp.vse.cz/pkdd99/Challenge/berka.htm (дата обращения: 24.05.2014).
17. Haas, P.J. A Bi-Level Bernoulli Scheme for Database Sampling. / P.J. Haas, C. Koenig // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13–18, 2004. — ACM, 2004. — P. 275–286.
18. Ioannidis, Y.E. Histogram-Based Approximation of Set-Valued Query-Answer. / Y.E. Ioannidis, V. Poosala // VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7–10, 1999, Edinburgh, Scotland, UK. — Morgan Kaufmann 1999. — P. 174–185.
19. John, G.H. Static Versus Dynamic Sampling for Data Mining. / G.H. John, P. Langley // Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD–96), Portland, Oregon, USA. — AAAI Press, 1996. — P. 367–370.
20. Lakshmi, M.S. Effectiveness of Parallel Joins. / M.S. Lakshmi, P.S. Yu // IEEE Transactions on Knowledge and Data Engineering. — 1990. — Vol. 2, No. 4. —P. 410–424.
21. Olken, F. Random Sampling from Database Files: A Survey. / F. Olken, D. Rotem // Statistical and Scientific Database Management, 5th International Conference SSDBM, Charlotte, NC, USA, April 3–5, 1990, Proceedings. — Springer, 1990. Lecture Notes in Computer Science. — P. 92–111.
22. Oracle Database SQL Language Reference. — URL: http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_10002.htm (accessed: 24.05.2014).
23. Palmer, C.R. Density Biased Sampling: an Improved Method for Data mining and Clustering. / C.R. Palmer, C. Faloutsos // Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16–18, 2000, Dallas, Texas, USA. — ACM, 2000. — P. 82–92.
24. Pan, C.S. Taming Elephants, or How to Embed Parallelism into PostgreSQL. / C.S. Pan, M.L. Zymbler // Database and Expert Systems Applications – 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26–29, 2013. Proceedings, Part I. — Springer, 2013. Lecture Notes in Computer Science. — Vol. 8055. — P. 153–164.
25. Parthasarathy, S. Efficient Progressive Sampling for Association Rules. / S. Parthasarathy // Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9–12 December 2002, Maebashi City, Japan. — IEEE, 2002. — P. 354–361.
26. Wu, X. Privacy Preserving Database Generation for Database Application Testing. / X. Wu, Y. Wang, S. Guo, Y. Zheng // Fundamenta Informaticae. — 2007. — Vol. 78, No. 1. — P. 595–612.

27. Yin, X. Efficient Classification across Multiple Database Relations: A CrossMine Approach. / X. Yin, J. Han, J. Yang, P.S. Yu // IEEE Transactions on Knowledge and Data Engineering. — 2006. — Vol. 18, No. 1. — P. 770–783.

Янцен Дмитрий Дмитриевич, магистрант кафедры системного программирования Южно-Уральского государственного университета (Челябинск, Российская Федерация), d.yantsen@gmail.com.

Цымблер Михаил Леонидович, к.ф.-м.н., доцент кафедры системного программирования Южно-Уральского государственного университета (Челябинск, Российская Федерация), mzym@susu.ru.

Гудков Владимир Юльевич, д.ф.-м.н., профессор кафедры ЭВМ Южно-Уральского государственного университета (Челябинск, Российская Федерация), diana@sonda.ru.

REPRESENTATIVE SAMPLING ALGORITHM FOR DATABASE SYSTEMS BASED ON THE PARTITIONED PARALLELISM

D.D. Yantsen, South Ural State University (Chelyabinsk, Russian Federation),
M.L. Zymbler, South Ural State University (Chelyabinsk, Russian Federation)
V.Yu. Gudkov, South Ural State University (Chelyabinsk, Russian Federation),

Sampling is a popular approach to very large databases processing in a wide range of applications, e.g. data mining, histograms construction, query execution cost estimation, etc. Use of either the sample instead of the original database can reduce the accuracy of the results, but offset by a reduction of time executing processing. Representative sampling allows you to save the sample of certain characteristics of the database. However, existing algorithms for representative sampling can not be used for pas-parallel database systems because it does not take into account the characteristics of the data distribution fissionable by the compute nodes of the cluster system. In this paper we propose al-representative sampling algorithm for parallel relational database systems based on the slice of parallelism. The results of computational experiments on the proposed algorithm, showing adequate maintenance of representativity database properties distributed across the nodes of a cluster system.

Keywords: relational databases, parallel database systems, representative sampling.

References

1. 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.
2. 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.
3. Sokolinsky L.B. Survey of Architectures of Parallel Database Systems // Programming and Computer Software. 2004. Vol. 30, No. 6. P. 337–346.

4. Yantsen D.D., Zymbler M.L. Algoritm reprezentativnogo semplinga dlya parallelnykh system baz dannykh [Representative sampling algorithm for parallel database systems] // Parallelnye vychislitelnye tekhnologii (PaVT'2014): Trudy mezhdunarodnoj nauchnoj konferentsii (Rostov-na-Donu, 1–3 aprelya 2014) [Parallel Computational Technologies (PCT'2010): Proceedings of the International Scientific Conference (Rostov-on-Don, Russia, April 1–3, 2014)]. Chelyabinsk, Publishing of the South Ural State University, 2014. P. 381.
5. Acharya S., Gibbons P.B., Poosala V., Ramaswamy S. Join Synopses for Approximate Query Answering // SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1–3, 1999, Philadelphia, Pennsylvania, USA. ACM, 1999. P. 275–286.
6. Agarwal S., Panda A., Mozafari B., Iyer A.P., Madden S., Stoica I. Blink and It's Done: Interactive Queries on Very Large Data // Proceedings of the VLDB Endowment. 2011. Vol. 5, No. 1. P. 1902–1905.
7. Bisbal J., Grimson J., Bell D.A. A Formal Framework for Database Sampling // Information & Software Technology. 2005. Vol. 47, No. 1. P. 819–828.
8. Buda T.S. Generation of Test Databases using Sampling Methods // International Symposium on Software Testing and Analysis, ISSTA'13, Lugano, Switzerland, July 15–20, 2013. ACM, 2013. P. 366–369.
9. Buda T.S., Cerqueus T., Kristiansen M., Murphy J. VFDS: Very Fast Database Sampling System // Proceedings of the IEEE 14th International Conference on Information Reuse & Integration, IRI 2013, San Francisco, CA, USA, August 14–16, 2013. IEEE, 2013. P. 153–160.
10. Buda T.S., Cerqueus T., Murphy J., Kristiansen M. CoDS: A Representative Sampling Method for Relational Databases // Database and Expert Systems Applications – 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26–29, 2013. Proceedings, Part I. Springer, 2013. Lecture Notes in Computer Science. Vol. 8055. P. 342–356.
11. Chakaravarthy V.T., Pandit V., Sabharwal Y. Analysis of Sampling Techniques for Association Rule Mining // Database Theory – ICDT 2009, 12th International Conference, St. Petersburg, Russia, March 23–25, 2009, Proceedings. ACM, 2009. P. 276–283.
12. Chaudhuri S., Das G., Srivastava U. Effective Use of Block–Level Sampling in Statistics Estimation // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13–18, 2004. ACM, 2004. P. 287–298.
13. Gemulla R., Rösch P., Lehner W. Linked Bernoulli Synopses: Sampling along Foreign Keys // Scientific and Statistical Database Management, 20th International Conference, SSDBM 2008, Hong Kong, China, July 9–11, 2008, Proceedings. Springer, 2008. Lecture Notes in Computer Science. P. 6–23.
14. Goethals B., Page W.L., Mampaey M. Mining Interesting Sets and Rules in Relational Databases // Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, March 22–26, 2010. ACM, 2010. P. 997–1001.
15. Gryz J., Guo J., Liu L., Zuzarte C. Query Sampling in DB2 Universal Database // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13–18, 2004. ACM, 2004. P. 839–843.

16. Guide to the Financial Data Set of the PKDD'99 Discovery Challenge. URL: http://lisp.vse.cz/pkdd99/Challenge/berka.htm (accessed: 24.05.2014).
17. Haas P.J., Koenig C. A Bi–Level Bernoulli Scheme for Database Sampling // Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13–18, 2004. ACM, 2004. P. 275–286.
18. Ioannidis Y.E., Poosala V. Histogram-Based Approximation of Set-Valued Query-Answer // VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7–10, 1999, Edinburgh, Scotland, UK. Morgan Kaufmann 1999. P. 174–185.
19. John G.H., Langley P. Static Versus Dynamic Sampling for Data Mining // Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD–96), Portland, Oregon, USA. AAAI Press, 1996. P. 367–370.
20. Lakshmi M.S., Yu P.S. Effectiveness of Parallel Joins // IEEE Transactions on Knowledge and Data Engineering. 1990. Vol. 2, No. 4. P. 410–424.
21. Olken F., Rotem D. Random Sampling from Database Files: A Survey // Statistical and Scientific Database Management, 5th International Conference SSDBM, Charlotte, NC, USA, April 3–5, 1990, Proceedings. Springer, 1990. Lecture Notes in Computer Science. P. 92–111.
22. Oracle Database SQL Language Reference. URL: http://docs.oracle.com/cd/E11882_01/server.112/e41084/statements_10002.htm (accessed: 24.05.2014).
23. Palmer C.R., Faloutsos C. Density Biased Sampling: an Improved Method for Data mining and Clustering // Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16–18, 2000, Dallas, Texas, USA. ACM, 2000. P. 82–92.
24. Pan C.S., Zymbler M.L. Taming Elephants, or How to Embed Parallelism into PostgreSQL // Database and Expert Systems Applications – 24th International Conference, DEXA 2013, Prague, Czech Republic, August 26–29, 2013. Proceedings, Part I. Springer, 2013. Lecture Notes in Computer Science. Vol. 8055. P. 153–164.
25. Parthasarathy S. Efficient Progressive Sampling for Association Rules // Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9–12 December 2002, Maebashi City, Japan. IEEE, 2002. P. 354–361.
26. Wu X., Wang Y., Guo S., Zheng Y. Privacy Preserving Database Generation for Database Application Testing // Fundamenta Informaticae. 2007. Vol. 78, No. 1. P. 595–612.
27. Yin X., Han J., Yang J., Yu P.S. Efficient Classification across Multiple Database Relations: A CrossMine Approach // IEEE Transactions on Knowledge and Data Engineering. 2006. Vol. 18, No. 1. P. 770–783.

УДК 004.4'22 DOI: 10.14529/cmse140403

ЭМУЛЯТОР PCI EXPRESS ДЛЯ HDL-МОДЕЛИРОВАНИЯ[1]

А.Б. Шворин

В данной работе описывается эмулятор PCI Express — инструмент, позволяющий упростить разработку и отладку некоторого класса аппаратных устройств, работающих по протоколу передачи данных PCI Express. Эмулятор позволяет промоделировать поведение разрабатываемого устройства на обычном компьютере, что значительно сокращает цикл отладки.

Ключевые слова: разработка аппаратного обеспечения, моделирование аппаратуры, эмуляция, PCI Express

Введение

Потребность в инструменте, позволяющем эмулировать работу PCI Express [1], была осознана автором и его коллегами в процессе работы над аппаратной реализацией интерконнекта в рамках проекта СКИФ-Аврора [2, 3] в 2010–2012 гг. Тогда же были сделаны первые наработки, которые в процессе развития выкристаллизовались в отдельный самостоятельный инструмент — эмулятор PCI Express. В дальнейшем в эмулятор была добавлена поддержка PCIe следующего поколения (Gen 3), что позволило с успехом применить его в следующем проекте Паутина [4].

В данной работе рассматривается задача разработки аппаратного дизайна для ПЛИС (программируемая логическая интегральная схема). Здесь ПЛИС может использоваться непосредственно в качестве целевой аппаратной платформы или же как средство быстрого прототипирования с целью дальнейшего переноса дизайна в специализированную микросхему.

Одним из факторов, негативно сказывающимся на продуктивности — общем времени разработки, — является длительный цикл отладки. Например, в проекте Паутина типичное время от внесения изменений в исходный текст дизайна до получения результатов теста составляет десятки минут, которые тратятся в основном на компиляцию и сборку дизайна. Разумным подходом является симуляция схемы в искусственной среде вместо ее загрузки в аппаратную среду (в ПЛИС). Такую возможность предоставляют современные САПР (системы автоматизированного проектирования, в данном случае специализированные для дизайна электронных устройств), в которых обычно ведется разработка, а также сторонние инструменты.

Основной сложностью такого подхода является то, что дизайн, как правило, не может работать сам по себе, он связан с какими-либо периферийными устройствами и требует от них выполнения определенного протокола. В случае загрузки дизайна в ПЛИС в роли периферии выступают устройства, электрически связанные с ножками ПЛИС. Если же происходит симуляция дизайна, то требуется каким-либо образом имитировать или, точнее, эмулировать поведение периферии. В простейших случаях (светодиодная индикация, кнопки, переключатели и др.) это можно делать средствами САПР, иногда вместо внешнего устройства достаточно реализовать простую заглушку и включить ее в основной дизайн, но бывают и гораздо более сложные устройства, так что их эмуляция представляет собой нетривиальную задачу.

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

В данной статье описывается способ реализации одного из ключевых компонентов модельной среды — виртуальной реализации интерфейса PCI Express. Этот эмулятор обладает достаточной гибкостью и поддерживает запуск стороннего программного обеспечения, работающего с устройством через PCI Express.

В разделе 1 кратко описана одна из реализаций интерфейса PCI Express — та, с которой работет эмулятор. Раздел 2 содержит общие идеи построения эмулятора в рамках клиент-серверной модели. Более подробное описание схемы работы серверной части эмулятора изложено в разделе 3. Там же показан механизм интеграции сервера с с пользовательским аппаратным дизайном. Наконец, в заключении статьи обозначена область применимости эмулятора и даны оценки его эффективности.

1. Интерфейс PCI Express

Интерфейс PCI Express (PCIe) является стандартом высокоскоростной передачи данных между различными устройствами компьютера, как правило, объединенными на одной плате. Для PCIe существуют также кабельные соединения. PCIe подразумевает наличие пакетного протокола адресной передачи данных, в котором имеется несколько типов пакетов, наиболее важные из которых перечислены ниже:

- запрос на запись по заданному адресу;
- запрос на чтение по заданному адресу;
- ответ на чтение.

Именно эти типы пакетов поддержаны в эмуляторе. Существуют также служебные с точки зрения протокола PCIe пакеты, но их поддержка в эмуляторе осталась невостребованной.

Поскольку проект Паутина подразумевал использование ПЛИС фирмы Altera, в качестве реализации интерфейса PCIe был взят предоставляемый этой фирмой протокол Avalon ST [5]. Таким образом, для эмулятора потребовалась программная реализация Avalon ST, что и было сделано. Разумеется, выбор конкретной реализации протокола сужает область применимости эмулятора, однако сравнительно легко добавляются новые протоколы на базе существующей библиотеки эмулятора.

На данный момент в эмуляторе реализованы следующие варианты протокола:

- 64-битный Avalon-ST;
- 128-битный Avalon-ST;
- 256-битный Avalon-ST без мультипакетного режима;
- 256-битный Avalon-ST в мультипакетном режиме.

Здесь разрядность означает количество данных, которое передается за один такт. Различные реализации PCIe могут иметь разную разрядность и работать и на разных частотах, что определяется поколением PCIe и количеством трансиверов. 256-битный вариант допускает так называемый мультипакетный режим (multiple packets per cycle), при котором на одном такте может передаваться хвост предыдущего пакета и начало следующего, что дает заметный выигрыш в пропускной способности по сравнению с обычным режимом.

Рассмотрим для примера 128-битный вариант Avalon-ST. Его HDL-интерфейс представлен на рис. 1.

Основными сигналами являются:

- rx_st_data — передаваемый поток данных, в данном случае разрядностью 128 бит;
- rx_st_sop, rx_st_eop — начало и конец пакета, соответственно;
- rx_st_valid — признак валидности данных на текущем такте.

```
entity ast128 is
    port (
        rx_st_sop   : in  std_logic;                        -- startofpacket
        rx_st_eop   : in  std_logic;                        -- endofpacket
        rx_st_err   : in  std_logic;                        -- error
        rx_st_valid : in  std_logic;                        -- valid
        rx_st_empty : in  std_logic;                        -- empty
        rx_st_ready : out std_logic;                        -- ready
        rx_st_data  : in  std_logic_vector(127 downto 0);   -- data
        rx_st_bar   : in  std_logic_vector(7 downto 0);     -- rx_st_bar
        ...
    );
end ast128;
```

Рис. 1. Интерфейс 128-битного варианта Avalon-ST

- rx_st_ready — сигнал готовности приемного устройства (в отличие от других сигналов, он является выходным для устройства).

Здесь представлена приемная (RX) часть интерфейса; передающая (TX) почти в точности такая же, но направления сигналов противоположны.

2. Принципы работы эмулятора PCI Express

Передача данных через эмулируемый PCIe осуществляется по клиент-серверной схеме, где роль сервера выполняет запущенный процесс собственно эмулятора, а клиентами являются обычные программы, работающие с устройством через PCIe.

Коммуникации между клиентом и сервером осуществляются двояко: через сокетный интерфейс Беркли [6] и посредством специального сегмента системной памяти, общего для клиента и для сервера. Сокетный интерфейс используется для инициализации клиента, а также в процессе работы для передачи PCI-запросов от клиента к серверу. PCI-запросы в обратном направлении (от сервера к клиенту) поступают напрямую в общий сегмент памяти, минуя сокетный механизм.

Технически клиентская часть эмулятора представляет собой библиотеку, с которой необходимо слинковать программу. Эта библиотека имеет интерфейс, названный mmdev, который представляет собой замену некоторым системным вызовам:

- mmdev_open() и mmdev_mmap() служат для инициализации клиента и дают ему доступ к общему сегменту системной памяти и к сегменту виртуальной памяти, связанному с устройством;
- mmdev_close() и mmdev_munmap() используются клиентом для завершения работы;
- mmdev_memcpy() заменяет стандартную функцию копирования памяти при обращении к памяти устройства, она генерирует PCI-запросы на чтение и запись.

Все перечисленные функции интерфейса mmdev имеют в точности те же прототипы, что и соответствующие им системные вызовы.

Для использования эмулятора обычная программа должна быть модифицирована следующим образом:

1. При запуске необходимо вызвать процедуру инициализации, которая обеспечит установку сокетного соединения с сервером.

2. Все операции доступа в сегмент памяти, связанный с устройством, должны быть заменены на специальный вызовы: вместо стандартного вызова memcpy() должен использоваться mmdev_memcpy(), а операция разыменования указателя должна быть выражена через него.

Здесь стоит отметить, что требование (2) является одной из основных трудностей на пути к использованию данного эмулятора. К сожалению, других способов перехвата обращений программы в память, по-видимому, нет. Например, рассматривалась идея использования механизма pagefault, но средства, предоставляемые операционной системой, оказались недостаточными для наших целей. С другой стороны, реализация библиотеки mmdev обеспечивает некоторую безопасность: если произошло прямое обращение по указателю в память эмулируемого устройства, то гарантированно произойдет segmentation fault, что позволит быстро локализовать ошибку и исправить ее путем замены обращения по указателю на вызов mmdev_memcpy().

Для удобства пользователя библиотека mmdev представлена в двух реализациях. Одна реализация, описанная выше, служит для работы с эмулятором. Другая — является тривиальной прослойкой между интерфейсом mmdev и соответствующими системными вызовами; она предназначена для работы с настоящим (не эмулируемым) устройством. Соответственно, есть разные способы использования библиотеки: слинковать пользовательскую программу с одной из реализаций статически или же сразу с обеими. Во втором случае выбор между реализациями будет осуществляться при запуске программы динамически, с помощью переменной среды.

3. Устройство сервера эмулятора

Двоякая реализация клиентской библиотеки обеспечивает выполнение важного принципа моделирования: виртуальная версия, предназначенная для симуляции, должна как можно меньше отличаться от реальной, которая загружается в ПЛИС. Реализация серверной части также следует этому принципу. А именно, подразумевается, что в дизайне устройства выделен модуль, интерфейс которого в точности совпадает с одной из версий представленного выше Avalon ST. Этот модуль-приложение, назовем его app, по сути является объектом тестирования, отладки и дальнейшей разработки. Все остальное — относительно неизменная обвязка, позволяющая проводить симуляцию и создавать прошивку для ПЛИС, то есть модуль допускает два способа использования.

Первый способ — это помещение модуля в «настоящий» дизайн верхнего уровня (toplevel design), предназначенный для загрузки в ПЛИС. В этом дизайне, разумеется, должен использоваться аппаратный модуль — адаптер PCIe из имеющихся в ПЛИС.

На рис. 2 показана работа некоторой пользовательской программы pgm (которая, вообще говоря, может быть запущена в виде не одного процесса, а нескольких) в аппаратном окружении, то есть когда в системе имеется ПЛИС с загруженным в нее дизайном. При этом обмен данными между процессами и устройствами полностью идет через PCIe.

Второй способ, симуляция — подразумевает агрегацию модуля в специальный дизайн верхнего уровня (toplevel design) с пустым интерфейсом (см. рис. 3). То, что интерфейс дизайна пустой, означает, что у него нет видимой периферии, и он не связан с внешней средой. Именно это и нужно для симуляции.

На рис. 4 представлена схема работы программы в режиме эмуляции. Для проведения симуляции должны быть запущены сервер (обозначен на схеме как процесс emu-server)

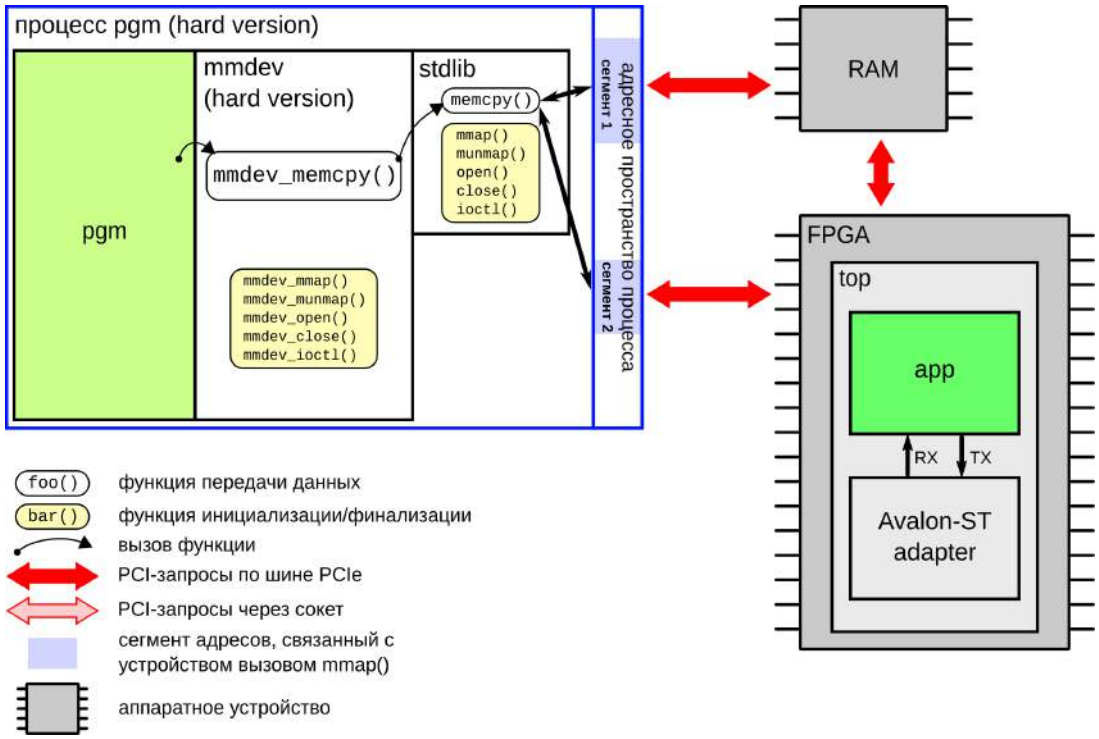

Рис. 2. Схема работы программы pgm с устройством app в обычном (аппаратном) режиме

```
entity emu_top is
end emu_top;
```

Рис. 3. Пустой HDL-интерфейс

и пользовательская программа (представлена процессом pgm), которая подсоединяется к серверу посредством сокетного интерфейса. Вообще говоря, пользовательских процессов может быть несколько, и все они будут клиентами сервера-эмулятора.

На этот раз обмен данными между процессорами и устройствами осуществляется иначе: доступ к памяти по-прежнему происходит через системную шину PCIe, а к эмулируемому устройству — через сокет. Таким образом реализуется вышеупомянутый принцип моделирования: при обоих режимах использования модуль-приложение app и пользовательская программа pgm остаются одними и теми же. Все изменения, вносимые в тестируемый модуль и в пользовательскую программу, в равной степени отражаются и при симуляции, и при запуске «в железе».

Для реализации симуляционной версии дизайна верхнего уровня был выбран GHDL [7] — открытая реализация компилятора VHDL, предназначенная только для симуляции. Важной особенностью GHDL, которая существенно используется в реализации эмулятора, является возможность привязки к методам, написанным на языке Си.

Реализация дизайна верхнего уровня с пустым интерфейсом emu_top является частью сервера. Ее устройство показано на рис. 5. Здесь ключевой особенностью является использование внешних (foreign) процедур, которые реализованы не на VHDL, а на Си. Они вызываются на каждом такте. В качестве параметров им передаются специальным образом запакованные сигналы Avalon ST: процедура line128_rx() обрабатывает приходящий в устройство поток (сигналы ast_rx_*), процедура line128_tx() обрабатывает, соответственно, исходящий поток (сигналы ast_tx_*). Параметры пакуются и передаются согласно спецификации

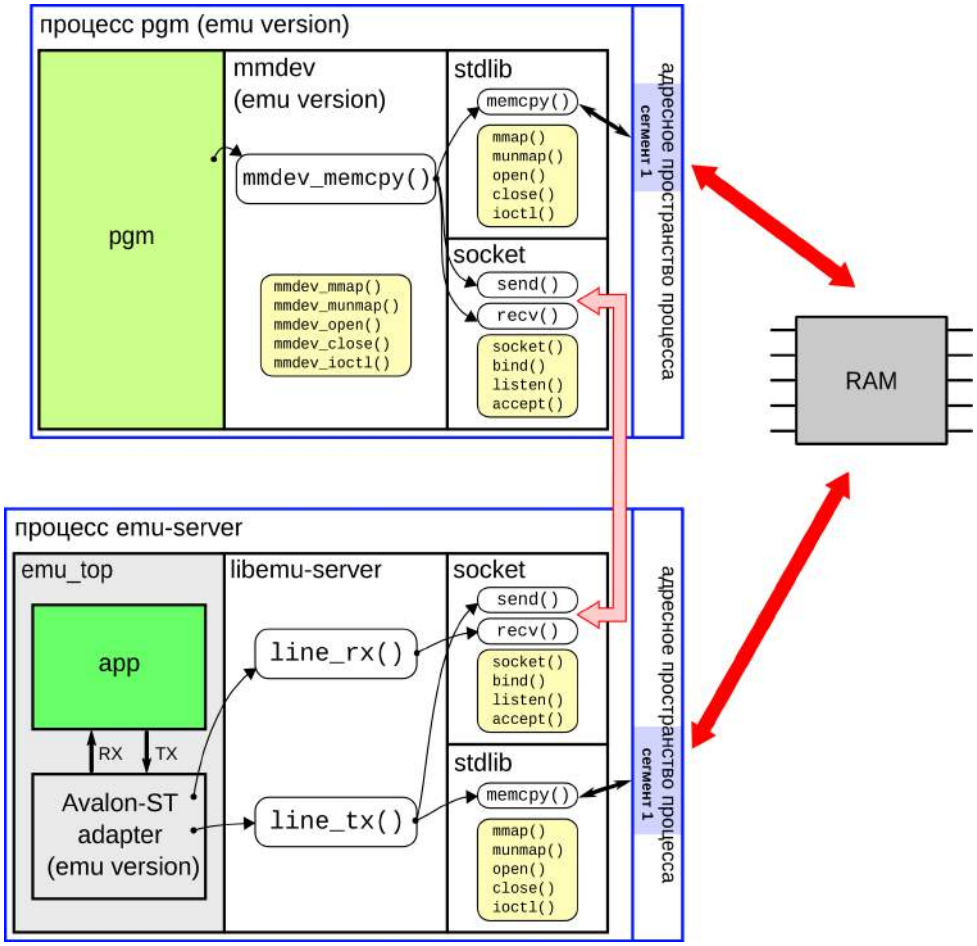

Рис. 4. Схема работы программы pgm с устройством app в режиме симуляции

GHDL [8]. Преобразование сигналов получается довольно громоздким, так что в приведенном листинге оно скрыто отточиями. Модуль-приложение, инстанциированный здесь как VHDL-компонент app, принимает поток PCI-запросов через сигналы ast_rx_* и выдает через ast_tx_*.

Реализация процедуры line128_tx() довольно проста: она в течение нескольких вызовов (один вызов за такт) аккумулирует части PCI-пакета и, когда наберется целый пакет, формирует вызов системной функции копирования memcpy(), имеющий следующий прототип:

```
void *memcpy(void *dest, const void *src, size_t n);
```

Целевой адрес dest и размер сообщения n для этого вызова получаются декодированием заголовка принятого PCI-пакета, а данные извлекаются из тела пакета. Нормальное поведение устройства подразумевает, что все данные направляются в общий сегмент памяти, где могут быть прочитаны клиентами. Для предотвращения опасного поведения, вызванного некорректной работой тестируемого устройства, желательно включить проверку целевого адреса.

В обратную сторону PCI-запросы передаются следующим образом. Вызов mmdev_memcpy() на клиенте передает свои аргументы на сервер через сокет. Там формируется PCI-пакет и помещается в специальную очередь, доступ к которой имеет процедура line128_rx(). Она, как было сказано выше, вызывается из VHDL-дизайна

```
architecture emu_top128 of emu_top is
    procedure line128_tx(...) is ...
    attribute foreign of line128_tx : procedure is "VHPIDIRECT line128_tx";

    procedure line128_rx(...) is ...
    attribute foreign of line128_rx : procedure is "VHPIDIRECT line128_rx";
    ...
begin
    process (clk, reset)
    begin
        if rising_edge(clk) then
            line128_rx(...);
            line128_tx(...);
        end if;
    end process;

    app : ast128
        port map (
            clk   => clk,
            reset => reset,
            ast_rx_... => ...,
            ast_tx_... => ...,
        );
    ...
end emu_top128;
```

Рис. 5. Реализация дизайна верхнего уровня 128-битного варианта Avalon-ST

верхнего уровня на каждом такте, и ее работа заключается в том, чтобы передать через свои параметры очередную часть PCI-пакета согласно протоколу Avalon ST.

Заключение

В режиме эмуляции цикл отладки очень короткий. Сборка занимает буквально несколько секунд — примерно столько же, как у среднего размера программы на языке Си, что несравнимо быстрее сборки прошивки для ПЛИС, которая обычно составляет порядка получаса. Скорость исполнения теста (симуляции) существенно зависит от сложности дизайна, и она, разумеется, на несколько порядков хуже, чем у аппаратной реализации. На практике редко требуется провести симуляцию больше чем на несколько тысяч тактов, что занимает от нескольких секунд до нескольких минут времени.

Таким образом, основное преимуществе симуляции — это, прежде всего, короткий цикл отладки. Также очевидное удобство в получении отладочной информации: работает отладочная печать VHDL, и можно сохранить и потом изучить всю трассу значений сигналов, в то время как для аппаратной реализации средства сохранения трассы весьма ограничены. На одной машине может быть запущено несколько независимых экземпляров сервера, к каждому из которых может присоединиться множество клиентов. Наконец, в режиме симуляции нет опасности привести машину, на которой проводится тестирование, в нерабочее состояние из-за генерации некорректных обращений в PCIe.

Следует также перечислить основные недостатки данного инструмента:

- необходимость имитации периферийных устройств помимо PCI Express;
- неполная эквивалентность реализации языка VHDL в компиляторе GHDL и САПР Quartus фирмы Altera, а также, вероятно, в САПР Xilinx ISE, что может потребовать небольшой модификации кода;
- отсутствие поддержки языка Verilog;
- неполная (на текущий момент) реализация протокола PCIe.

Работа над эмулятором продолжается в настоящее время. Его исходные коды доступны под свободной лицензией [9]. По мере необходимости планируется добавить поддержку различных версий протокола Avalon ST, а также обеспечить более полное покрытие функционала PCIe. Автор выражают надежду, что данный инструмент окажется удобным и достаточно универсальным и сможет найти новых пользователей.

Работа выполнена при поддержке Программы фундаментальных исследований Президиума РАН № 18.

Литература

1. IP Compiler for PCI Express User Guide URL: `http://www.altera.com/literature/ug/ug_pci_express.pdf` (дата обращения: 26.08.2014)

2. Абрамов, С.М. Возможности суперкомпьютеров «СКИФ» ряда 4 по аппаратной поддержке в ПЛИС различных моделей параллельных вычислений / С.М. Абрамов, С.А. Дбар, А.В. Климов, Ю.А. Климов, А.О. Лацис, А.А. Московский, А.Ю. Орлов, А.Б. Шворин // Суперкомпьютерные технологии: разработка, программирование, применение (СКТ-2010): Материалы международной научно-технической конференции (Дивноморское, 27 сентября – 2 октября 2010). — Таганрог: Изд-во ТТИ ЮФУ, 2010. — Том 1. — С. 11–21.

3. Абрамов, С.М. Суперкомпьютеры «СКИФ» ряда 4 / С.М. Абрамов, В.Ф. Заднепровский, Е.П. Лилитко // Информационные технологии и вычислительные системы. — 2012. — № 1. — С. 3–16.

4. Абрамов, С.М. О разработке интерконнекта на активных оптоволоконных кабелях и программируемых логических интегральных схемах / С.М. Абрамов, И.А. Адамович, С.А. Блохин, А.В. Елистратов, Л.Я. Карачинский, Ю.А. Климов, И.И. Новиков, А.Ю. Пономарев, С.С. Ранцев, И.А. Фохт, А.Ю. Хренов, А.Б. Шворин, Ю.В. Шевчук // Научный сервис в сети Интернет: все грани параллелизма: Труды Международной суперкомпьютерной конференции (Новороссийск, 23–28 сентября 2013). — М.: Изд-во МГУ, 2013. — С. 220–223.

5. Avalon Interface Specifications URL: `http://www.altera.com/literature/manual/mnl_avalon_spec.pdf` (дата обращения: 26.08.2014)

6. BSD Sockets Interface Programmer's Guide URL: `http://www.cs.put.poznan.pl/wswitala/download/pdf/B2355-90136.pdf` (дата обращения: 26.08.2014)

7. GHDL guide URL: `http://ghdl.free.fr` (дата обращения: 26.08.2014)

8. GHDL Restrictions on foreign declarations URL: `http://ghdl.free.fr/ghdl/Restrictions-on-foreign-declarations.html` (дата обращения: 26.08.2014)

9. Репозиторий исходных кодов эмулятора PCI Express URL: `https://github.com/shvorin/pcie-emu` (дата обращения: 26.08.2014)

Шворин Артем Борисович, инженер-программист Института программных систем им. А.К. Айламазяна Российской академии наук (Переславль-Залесский, Российская Федерация), art@shvorin.net.

Поступила в редакцию 25 августа 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 51–60

DOI: 10.14529/cmse140403

PCI EXPRESS EMULATOR FOR HARDWARE DESIGN MODELLING

A.B. Shvorin, Program Systems Institute of RAS (Pereslavl-Zalessky, Russian Federation)

This paper describes PCI Express emulator. This tool is aimed to simplify development and debugging of certain class hardware devices using PCI Express protocol. The emulator is capable to simulate a device under development by a conventional computer. It significantly reduces debug time.

Keywords: hardware design, hardware simulation, emulation, PCI Express

References

1. IP Compiler for PCI Express User Guide URL: `http://www.altera.com/literature/ug/ug_pci_express.pdf` (accessed: 26.08.2014)
2. Abramov S.M., Dbar S.A., Klimov A.V., Klimov Yu.A., Lacis A.O., Moskovskij A.A., Orlov A.Yu., Shvorin A.B. Vozmozhnosti superkomp'yuterov "SKIF" ryada 4 po apparatnoj podderzhke v PLIS razlichnyx mod elej parallel'nyx vychislenij [The Capability of "SKIF" Grade 4 Supercomputers to Hardware Support of Various Parallel Evaluation Models in FPGA] // Superkomp'yuternye texnologii: razrabotka, programmirovanie, primenenie (SKT-2010): Materialy mezhdunarodnoj nauchno-texnicheskoj konferencii (Divnomorskoe, 27 sentyabrya – 2 oktyabrya 2010) [Supercomputing Technology: Development, Programming, Applying: Proceedings of the International Scientific and Technical Conference (Divnomorskoe, Russia, September, 27 – October, 2, 2010)]. Taganrog, Publishing of Taganrog Technology Institute of the South Ural State University. 2010. Vol. 1. P. 11–21.
3. Abramov S.M., Zadneprovskij V.F., Lilitko E.P. Superkomp'yutery "SKIF" ryada 4 ["SKIF" grade 4 Supercomputers] // Informacionnye texnologii i vychislitel'nye sistemy [Information Technology and Computer Systems]. 2012. No 1. P. 3–16.
4. Abramov S.M., Adamovich I.A., Bloxin S.A., Elistratov A.V., Karachinskij L.Ya., Klimov Yu.A., Novikov I.I., Ponomarev A.Yu., Rancev S.S., Foxt I.A., Xrenov A.Yu.,

Shvorin A.B., Shevchuk Yu.V. O razrabotke interkonnekta na aktivnyx optovolokonnyx kabelyax i programmiruemyx logicheskix integral'nyx sxemax [The Development of Interconnect Based on Active Optical Cables and FPGA] // Nauchnyj servis v seti Internet: vse grani parallelizma: Trudy Mezhdunarodnoj superkomp'yuternoj konferencii (Novorossijsk, 23–28 sentyabrya 2013) [Scientific Research in the Internet: All the Faces of Parallelizm: Proceedings of the International Supercomputer Conference (Novorossiysk, Russia, September, 23–28, 2013)]. Moscow, Publishing of MSU. 2013. P. 220–223.

5. Avalon Interface Specifications URL: `http://www.altera.com/literature/manual/mnl_avalon_spec.pdf` (accessed: 26.08.2014)
6. BSD Sockets Interface Programmer's Guide URL: `http://www.cs.put.poznan.pl/wswitala/download/pdf/B2355-90136.pdf` (accessed: 26.08.2014)
7. GHDL guide URL: `http://ghdl.free.fr` (accessed: 26.08.2014)
8. GHDL Restrictions on foreign declarations URL: `http://ghdl.free.fr/ghdl/Restrictions-on-foreign-declarations.html` (accessed: 26.08.2014)
9. PCI Express emulator source code repository URL: `https://github.com/shvorin/pcie-emu` (accessed: 26.08.2014)

Received August 25, 2014.

УДК 004.627, 004.656 DOI: 10.14529/cmse140404

АЛГОРИТМ ФРАКТАЛЬНОГО ПОИСКА В РЕЛЯЦИОННЫХ БАЗАХ ДАННЫХ

Т.Ю. Лымарь, Т.С. Мантрова, Н.Ю. Староверова

Статья посвящена вопросам разработки алгоритмов фрактального анализа реляционных баз данных. Дается обзор и сравнительный анализ известных приложений теории фракталов к обработке данных. Предложен новый алгоритм фрактального поиска в реляционной базе данных, позволяющий обнаруживать повторяющиеся группы данных. Приведена общая схема алгоритма. Рассмотрена реализация для СУБД Oracle. Представлена реализация с использованием модели распределенных вычислений MapReduce. Приводятся примеры использования разработанного алгоритма для сжатия и анализа содержимого базы данных

Ключевые слова: реляционные базы данных, теория фракталов, фрактальный анализ баз данных, сжатие данных.

Введение

Корпоративная база данных любого современного предприятия обычно содержит набор таблиц, хранящих огромное количество записей о тех или иных фактах либо объектах. Как правило, каждая запись в подобной таблице описывает какой-то конкретный объект или факт и, как правило, структуры этих записей идентичны. В связи с тем, что современные базы данных содержат огромное количество данных, которые необходимо не только компактно хранить, но и анализировать, осуществлять поиск. Для извлечения необходимой информации разрабатываются разнообразные теории и алгоритмы. Одной из подобных теория является фрактальный анализ.

«Все фигуры, которые я исследовал и называл фракталами, в моем представлении обладали свойством быть нерегулярными, но самоподобными», — писал Бенуа Мандельброт, который в 1975 г. ввел термин фрактал (от латинского fractus — дробный) [16]. Такое определение позволяет охватить широкое множество объектов, которые подпадают под понятие фрактала. Применение теории фракталов в различных областях сводится к поиску самоподобных простых частей (*доменов*), применив к которым определенную итерационную функцию, можно получить всю систему. В информационных технологиях наиболее распространенным вариантом применения теории фракталов является ее приложение к графической информации [19], что вполне естественно и понятно, однако можно рассмотреть с точки зрения данной теории и системы баз данных. Совокупность большого количества записей таблиц, может стать источником дополнительной, гораздо более ценной информации, которую нельзя получить на основе одной конкретной записи. Вполне возможно, что анализируемая информация может быть самоподобна и может отражать определенную зависимость не только между записями таблиц, но и внутри самой записи. Подобного рода информация обычно используется при прогнозировании, планировании, анализе, и ценность ее очень высока, поэтому выявление структуры данных — ключевой аспект эффективного представления и хранения этих данных.

Целью данного исследования является рассмотрение реляционных таблиц с точки зрения теории фракталов, определение подходящих методов фрактального анализа и разработка алгоритма поиска доменов в реляционных таблицах. Кроме того, необходимо

продемострировать применение полученных результатов, например, при поиске функциональных зависимостей в таблицах и сжатии хранимых данных.

В разделе 1 представлен обзор фрактальных методов интеллектуального анализа данных. Раздел 2 посвящен построени. алгоритма поиска самоподобных частей в реляционной таблице. Раздел 3 описывает ключевые аспекты реализации разработанной системы. В разделе 4 представлены результаты экспериментов для определения эффективности разработанного алгоритма. В заключении суммируются основные результаты работы.

1. Обзор методов Fractal Data Mining

Технология Data Mining основана на статистических методах и служит для выявления в «сырых» данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретаций знаний, необходимых для принятия решений. Применение фрактальных преобразований и самоподобия также относится к методам интеллектуального анализа данных [1]. Наиболее часто данная техника используются в алгоритмах сжатия данных без потери информации. Рассмотрим некоторые методы интеллектуального анализа данных, которые могут реализовываться с помощью фрактальных техник.

Кластеризация выполняет разбиение элементов некоторого множества на группы в зависимости от их схожести. У кластеризации существует большое количество практических применений, как в информатике, так и в других областях. Примерами применения могут служить: анализ данных, извлечение и поиск информации, группировка и распознавание объектов. Также кластеризация сама по себе является важной формой абстракции данных [17].

Использование самоподобия в кластеризации обеспечивает очень естественный способ определения кластеров и не ограничивается какой-либо конкретной формой кластера. В работе [2] проведен эксперимент с использованием алгоритма фрактальной кластеризации, основанный на самоподобии свойств наборов данных в кластере точек. Данный алгоритм назван «Фрактальная кластеризация» (ФК), постепенно определяет место точки в кластере, для которого может происходить изменение фрактальной размерности после добавления точки. Это очень естественный способ кластеризации точек, так как точки в одном кластере имеют большую степень самоподобия чем вне него. При добавлении новых точек вычисляется новая фрактальная размерность кластера. При этом уже образованные кластеры могут разбиваться или соединяться [2]. В работе [10] предложены алгоритмы обработки изображений, которые позволяют избежать проблем многократного повторения сканирования больших наборов данных. В работе [3] предложен метод кластеризации многомерных массивов данных с использованием MapReduce. Основные рассмотренные задачи: минимизация затрат ввода и вывода, способ разделения точек данных и объединение результатов.

Понижение размерности понимается как исключение коррелирующих атрибутов отношения. Атрибуты, которые могут быть рассчитаны из других по известному методу, можно исключить, применив методы Data Mining без ущерба для результатов [11]. Таким образом, задача превращается в выявление корреляции между атрибутами в наборе данных, и подсчет количества избыточных атрибутов, находящихся в наборе. Данный подход можно рассматривать как способ сжатия: рассматривать только те атрибуты, которые поддерживают основные характеристики хранящихся данных.

Для выявления повторяющихся данных из информации используется такое свойство «фрактальная размерности» (ФР) данных. ФР, по сути, есть оценка степени свободы набора данных. Она дает нам представление о том, каким образом данные распространяются в пространстве данных. Использование ФР набора данных может сократить время анализа данных. Основная идея, изложенная в статье [8], заключается в использовании ФР данных, и отказе от атрибутов, которые не влияют на нее. ФР является относительно устойчивой к воздействию избыточных атрибутов. Таким образом, предлагается своего рода алгоритм обратной ликвидации, который использует быстрое вычисление ФР. Этот алгоритм последовательно удаляет атрибуты, которые способствуют минимуму на ФР. В работе [8] представлен быстрый, масштабируемый алгоритм для быстрого выбора наиболее важных атрибутов (размеров) для данного набора n-мерных векторов.

2. Алгоритм фрактального поиска в реляционной БД

Мандельброт о своей теории фракталов отозвался так: «Рискнув ответить на вызов, я задумал и разработал новую геометрию природы, а также нашел для нее применение во многих разнообразных областях. Новая геометрия способна описать многие из неправильных и фрагментированных форм в окружающем нас мире и породить вполне законченные теории, определив семейство фигур, которые я называю фракталами. Наиболее полезные фракталы включают в себя элемент случайности; как правильность, так и неправильность их подчиняется статистическим законам» [16].

Покажем, что реляционная таблица также является фракталом. Для начала приведем основные определения теории реляционных баз данных. Схемой отношения R называется конечное множество имен атрибутов $\{A_1, A_2, \ldots, A_n\}$. Каждому имени атрибута A_i ставится в соответствие множество D_i — множество значений атрибута A_i, $1 \le i \le n$, D — объединение D_i, $1 \le i \le n$. Отношение r со схемой R — это конечное множество отображений $\{t_1, t_2, \ldots, t_n\}$ из R в D; причем каждое отображение t должно удовлетворять следующему ограничению: $t(A_i)$ принадлежит D_i, $1 \le i \le n$. Эти отображения называются кортежами [12].

Из определения отношения r можно отметить, что кортеж — это составная часть, которая несет в себе основную информацию о структуре отношения. Добавляя каждый последующий кортеж, мы строим отношение.

С точки зрения фрактального анализа доменом для отношения может выступать кортеж целиком или некоторая его часть. Таким образом, доменом может являться некоторая комбинация атрибутов одного кортежа, тем самым мы можем разложить отношение на еще более мелкие части, сохранив при этом сведения о его структуре.

Каждый атрибут ограничен определенным множеством значений, значит, и пара атрибутов будет также ограничена некоторым множеством. Тогда можно предположить, что домены в отношении могут быть не только идентичными по структуре, но и иметь одинаковые значения (рис. 1).

Стоит отметить тот факт, что первичный ключ отношения может выступать как некоторая функция, которая позволит определить соответствие между значением домена и кортежем.

Имя_Д	**Город_Д**	**Объем_Д**	**Цвет_Д**
Гайка	Тула	5	Красный
Гайка	Тула	10	Синий
Болт	Сургут	5	Красный
Болт	Сургут	10	Синий
Болт	Сургут	20	Черный

Болт	Сургут
Болт	Сургут

5	Красный
10	Синий
20	Черный

Рис. 1. Домены на уровне значений

Таким образом, выделяя домены в отношении, можно определить взаимосвязь атрибутов и фрактально закодировать таблицу, сохраняя сведения о ее структуре. Что позволит не только извлечь новые полезные и нетривиальные знания, но и представить отношение в более компактном виде. Полученная информация может использоваться при прогнозировании, планировании, анализе, и ценность ее очень высока, поэтому выявление структуры данных — ключевой аспект эффективного представления и хранения этих данных.

Основная идея алгоритма фрактального поиска сформулирована в [15] и заключается в подборе такого множества доменов, которое полностью опишет структуру отношения и позволит представить его содержимое минимально возможным количеством записей.

В общем виде алгоритм можно представить следующим образом (рис. 2):

Рис. 2. Общий вид алгоритма поиска доменов

Каждая часть алгоритма представляет собой отдельную задачу, которую можно решить различными способами. Далее опишем каждый шаг общего алгоритма поиска доменов более подробно.

2.1. Список возможных структур доменов

Как было сказано, доменом мы будем считать некоторую комбинацию атрибутов одного кортежа. Размером домена мы будем называть количество входящих в него атрибутов.

Пусть n — это количество атрибутов в таблице, а k — это размер домена, причем $1 \le k \le n$. Тогда количество возможных доменов размера k в таблице составляет:

$$C_n^k = \frac{n!}{k!(n-k)!}$$

Тогда общее количество всевозможных структур доменов будет:

$$\sum_{k=1}^{n} C_n^k = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!}$$

При фрактальном кодировании изображения возникал вопрос об оптимизации поиска и подбора доменных областей таким образом, чтобы сохранить точность изображения. В результате под структурой домена в задаче сжатия изображения рассматривался не только размер домена, но и его вид: прямоугольник, треугольник, шестиугольник и т.д. В силу того, что в отношении кортеж есть конкретный объект, то можно считать, что иное определение домена для отношения не представляется подходящим. Таким образом, говоря о структуре домена, мы ограничиваемся лишь его размером. Оценим примерное количество всех возможных структур доменов отношения. При формировании отношения количество входящих в него атрибутов стараются ограничивать 15, поэтому количество возможных структур доменов не должно превышать 32767. Однако если при проектировании схемы базы данных не придерживаться данного ограничения, то количество атрибутов в таблице может быть намного больше, следовательно, резко возрастет количество возможных структур доменов. Тем самым сложность во многом зависит от количества атрибутов в таблице.

В связи с описанной выше проблемой, возникает вопрос о времени поиска и достаточном множестве структур доменов, которые позволят наиболее полно представить описываемую систему. Основываясь на статистических сведениях о данных, содержащихся в базе, становится возможным сократить список структур доменов. Чем меньше количество рассматриваемых структур, тем меньше времени необходимо на поиск различных значений и на формирование оптимального множества доменов.

2.2. Количество различных значений домена

Количество различных значений домена является важнейшей характеристикой структуры, по сути, она является критерием того, насколько данная структура нам подходит. Чем меньше количество различных значений домена, тем лучше, тем больше информации о взаимосвязи атрибутов структура домена отображает.

Имея сведения о количестве различных значений атрибутов, мы всегда можем максимально оценить количество различных значений для произвольной структуры домена. Количество различных значений домена всегда меньше или равно произведения количеств различных значений атрибутов, входящих в домен. Обозначим оценку количества различных значений домена KD. Значение KD может быть как больше количества кортежей в таблице, так и меньше. Если KD окажется намного больше количества кортежей в отношении, то структуру, количество различных значений которой ограничивает данное значение, рассматривать смысла не имеет. Однако если KD окажется намного меньше

количества кортежей в отношении, то данная структура, возможно, позволит подобрать оптимальное множество для кодирования отношения или разбить объекты таблицы на более крупные классы.

Исходя из изложенных выше утверждений, не вполне очевидно, что именно может являться ограничением величины KD. Выбор можно осуществлять в зависимости от поставленной задачи и максимально допустимого для нее количества различных значений. Например, сжатие в определенное число раз потребует выбора параметра KD, который будет равен части количества кортежей таблицы. Так, чтобы рассмотреть все возможные домены, значение параметра KD не должно превышать половины количества всех кортежей отношения. Если же решается задача сжатия отношения, то значение параметра KD можно ограничить меньшими значениями, например $1/3$ или $1/4$ от мощности отношения. Поведение алгоритма при различных значениях параметра KD будет исследовано в экспериментах.

Алгоритм поиска точного количества различных значений весьма прост: мы просто сравниваем каждую считанную комбинацию с уже имеющимися значениями.

Поиск количества различных значений домена является самой затратной частью алгоритма. Поэтому очень важно заранее отбросить некоторое количество структур доменов, чтобы сократить время выполнения данной части. Исключение неподходящих структур доменов может проводиться на основании сведений, хранящихся в словаре базы данных. Оценивая количество различных значений доменов, можно исключить те структуры, которые содержат большое число отличных данных. Кроме того, алгоритм поиска количества различных значений легко поддается распараллеливанию на всем множестве структур доменов, так как для каждой структуры домена алгоритм может выполняться независимо.

2.3. Оптимальное множество доменов

После того, как было получено количество различных значений доменов, можно сформировать множество доменов, которые полностью описывают таблицу. *Оптимальным множеством доменов* D будем называть совокупность структур доменов $D = D_1 \cup ... \cup D_m$, которое содержит в совокупности все атрибуты таблицы в единственном экземпляре, и сумма количества различных значений доменов будет минимальна.

Алгоритм поиска оптимального множества доменов можно представить в виде рекурсивной функции, последовательно добавляющей новые домены до тех пор, пока не просмотрены все атрибуты и общее количество различных значений доменов не превышает текущего минимума. При первом запуске значение минимума определяется как *(количество атрибутов в отношении)*(количество строк в отношении)*.

Найденное оптимальное множество доменов позволит получить список различных значений, которых достаточно, чтобы восстановить таблицу, задав определенную систему функций.

Заметим, что не для всех структур доменов количество различных значений может быть существенно меньше количества кортежей в отношении. Однако, получив сведения о количестве различных значений доменов, можно сформировать алгоритмы для извлечения новых, нетривиальных знаний из рассматриваемой таблицы:

- Понижение размерности. Полученный набор различных значений позволит определить наличие функциональных зависимостей в таблице, тем самым появляется возможность вычислить коррелирующие атрибуты.
- Классификация. Основываясь на структурах доменов с меньшим количеством различных значений, можно явно выделить классы, на которые разбиваются объекты в таблице.
- Кластеризация. Выделение доменов позволяет выполнять последовательную кластеризацию объектов, тем самым появляется возможность некоторого регулирования размера кластера и признаков разбиения, то есть разместить объект в кластере, основываясь на некоторой значимой части.

Основываясь на полученных количествах различных значений доменов, становится возможным не только закодировать отношение, но и получить достаточно информации о зависимостях между атрибутами. Основной сложностью данной части алгоритма является большое количество полученных структур доменов. Чем больше структур, тем больше информации необходимо обработать и проанализировать, поэтому время выполнения последней части во многом зависит от мощности списка возможных структур доменов.

3. Реализация алгоритма

Для реализации программной системы, был выбран язык программирования Java. Определим основные объекты, которые определяют основную логику программной системы и, соответственно, реализуют описанные выше алгоритмы. Диаграмма классов представлена на рис. 3.

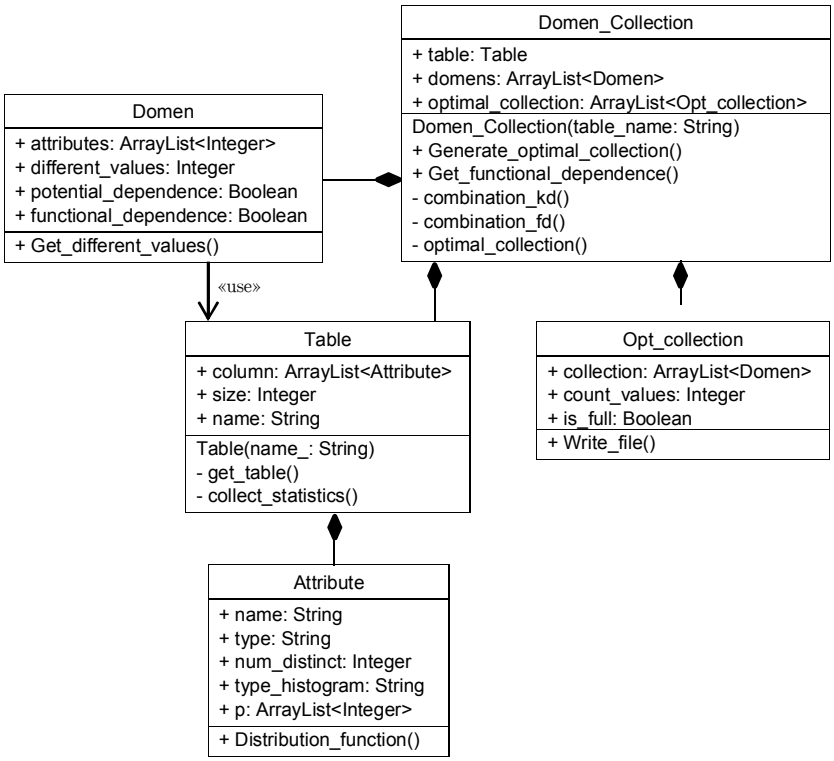

Рис. 3. Диаграмма классов

Класс «Atribute» описывает столбец отношения. Класс содержит набор атрибутов: имя столбца; тип данных; тип гистограммы; функция распределения, и метод построения функции распределения. Данные о распределении значений в столбце извлекаются из словаря СУБД. В качестве СУБД в данной реализации была использована Oracle 10g. Словарь СУБД Oracle содержит огромное количество таблиц с различной информацией о структуре базы данных, настройках системы [13], в том числе необходимые нам инструменты для сбора статистической информации и таблицы словаря данных, в которых собирается информация о распределении значений в столбцах таблицы: количество различных значений и гистограмма.

Класс «Table» задает структуру, согласно которой будут храниться сведения об анализируемом отношении. Класс содержит набор атрибутов: список атрибутов отношения; мощность отношения; имя таблицы, и методы построения статистики и получения отношения, реализующие взаимодействие с СУБД Oracle.

Класс «Domen» описывает структуру наименьшего основного фрактального объекта — домена. Атрибутами данного класса являются: список индексов, которые являются ссылками на атрибуты таблицы, составляющих домен; количество различных значений; признак наличия потенциальной функциональной зависимости; признак наличия функциональной зависимости. У класса только один метод — подсчет количества различных значений.

Класс «Domen_Collection» реализует основные алгоритмы для получения списка доменов, который позволит построить оптимальное множество и выделить функционально зависимые атрибуты. Атрибутами данного класса являются: список доменов; объект «Table», список оптимальных множеств (так как разные наборы доменов можеут давать одинаковое количество записей). Интерфейсные методы: определения функциональных зависимостей и построение оптимального множества.

Класс «Opt_collect» задает структуру для хранения оптимального множества. Класс содержит набор атрибутов: список доменов, составляющих множество; количество записей; признак полного заполнения, и метод записи хранящего оптимального множества в файл.

Как уже отмечалось выше, этап поиска количества различных значений домена является наиболее вычислительно сложным — порядка $O(m)$, где m — мощность отношения. Поэтому для реализации этого этапа алгоритма была выполнена дополнительная параллельная реализация с использованием технологии распределенных вычислений MapReduce [7]. В качестве платформы для реализации выбран проект с открытым исходным кодом Hadoop [9].

В соответствии с принципами концепции MapReduce [4] каждый домен обрабатывается Map-элементами. Пары <ключ, значение>, произведенные всеми Map-элементами, группируются по ключу и направляются в виде пары <ключ, массив значений> в адрес Reduce-элементов. Подсчитывается количество Reduce-элементов и выдается окончательный результат [14]. На рис. 4 показан пример обработки домена.

Полученное количество различных значений доменов служит основой для выбора оптимального множества доменов, которое полностью описывает таблицу.

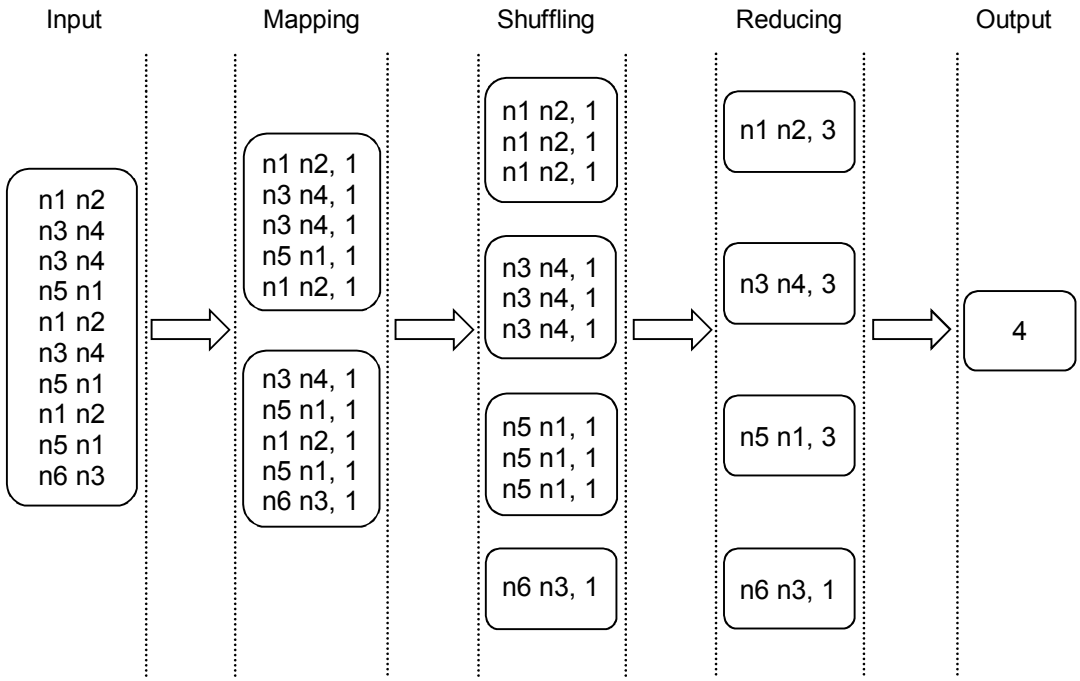

Рис. 4. Пример подсчета количества различных значений домена

Построенный алгоритм поиска оптимального множества можно отнести к алгоритму сжатия без потерь, тогда как обычно алгоритмы фрактального сжатия относят к алгоритмам с потерей информации. Сжатие с потерей информации для баз данных неприменимо. Немаловажным является и тот факт, что сжатую таким образом таблицу можно использовать для работы: выполнения запросов, вставки и удаления информации.

4. Эксперименты

Как было отмечено ранее, в отличие от большинства конкурирующих методов анализа, в которых большие наборы данных являются проблемой, метод на фрактальной основе страдает только тогда, когда данных слишком мало. Поэтому в качестве тестовой базы данных будут рассмотрены большие объемы данных реального программного комплекса «единый государственный реестр земель» (далее ПК ЕГРЗ) [18], предназначенный для ведения государственного земельного кадастра на уровне кадастрового района. Структура базы данных ПК ЕГРЗ менялась в зависимости от изменений в законе, появления приказов. Постоянное внесение изменений в структуру базы данных привело к появлению функциональных зависимостей и увеличению количества атрибутов в таблице. В качестве тестовых отношений будут рассмотрены таблицы OBJLOT, содержащая сведения о характеристиках земельных участков, и OBJ, содержащая список объектов. Таблица OBJLOT состоит из 53 атрибутов, количество кортежей 10568. Таблица OBJ состоит из 16 атрибутов, количество кортежей 63636. Таблицы заполнены сведениями, отраженными на публичной кадастровой карте [17].

Над разработанной системой было проведено несколько различных серий экспериментов. В данной статье рассмотрим два основных блока:

1) исследование эффективности кодирования таблицы оптимальным множеством доменов,
2) анализ быстродействия параллельной реализации.

Для проведения первого блока экспериментов был использован следующий подход: если сохранить в простой текстовый файл таблицу, закодированную с помощью оптимального множества доменов, и таблицу с полными записями, то сравнением размеров файлов будет получен коэффициент сжатия информации, содержащейся в таблице.

Рассмотрим построение оптимального множества для таблицы OBJ. Сравним размеры закодированных файлов при различных значениях параметра KD (табл. 1).

Таблица 1

Результаты сжатия таблицы OBJ

№ п/п	Содержимое файла	Размер файла	Коэффициент сжатия
1	Исходная таблица	5,02 Mb	1
2	Закодированная таблица, $KD = 25$ % строк таблицы	1,9 Mb	2,7
3	Закодированная таблица, $KD = 33$ % строк таблицы	1,9 Mb	2,7
4	Закодированная таблица, $KD = 50$ % строк таблицы	1,5 Mb	3,34

В результате получается, что в лучшем случае мы сможем сжать таблицу OBJ в 3,34 раза, в худшем — в 2,7 раз.

Рассмотрим построение оптимального множества для таблицы OBJLOT. Аналогично сравним размеры закодированных файлов (табл. 2).

Таблица 2

Результаты сжатия таблицы OBJLOT

№ п/п	Содержимое файла	Размер файла	Коэффициент сжатия
1	Исходная таблица	3,25 Mb	1
2	Закодированная таблица, $KD = 25$ % строк таблицы	1,3 Mb	2,50
3	Закодированная таблица, $KD = 33$ % строк таблицы	1,25 Mb	2,60
4	Закодированная таблица, $KD = 50$ % строк таблицы	1,23 Mb	2,64

В результате получается, что в лучшем случае мы сможем сжать таблицу $OBJLOT$ в 2,64 раза, в худшем — в 2,5 раза.

Таким образом, очевидно, что представление таблицы базы данных посредством оптимального множества доменов весьма эффективно.

В заключение рассмотрим результаты экспериментов над параллельной версией алгоритма. Эксперименты проводились узле суперкомпьютера «СКИФ-Аврора», характеристики которого приведены в табл. 3.

Таблица 3

Аппаратная платформа экспериментов

Характеристика	Значение

Число выч. процессоров/ядер	2/12
Тип процессора	Intel Xeon X5680 (Gulftown, 6 ядер по 3,33 ГГц)
Оперативная память	12 Гб
Тип управляющей сети	InfiniBand QDR (40 Гбит/с, макс. задержка 2 µs)

Была исследована зависимость времени выполнения алгоритма от количества используемых узлов. График ускорения выполнения алгоритма приведен на рис. 5.

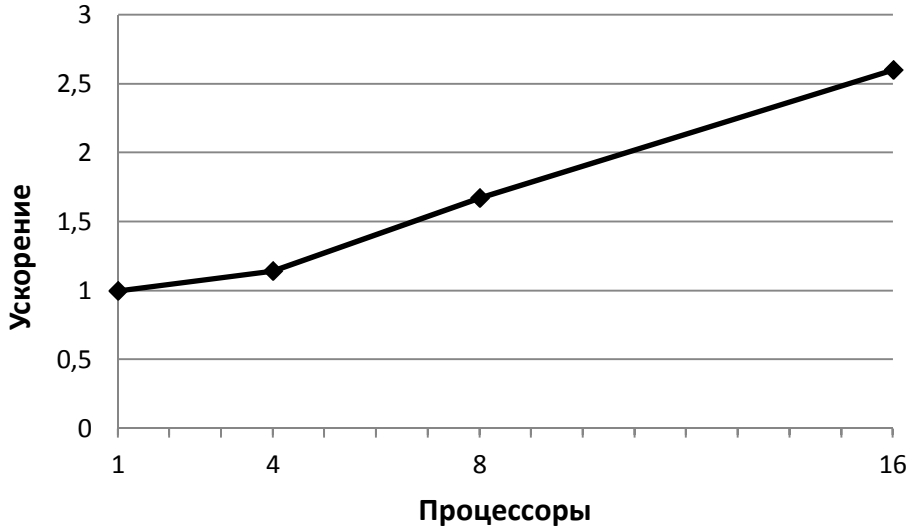

Рис. 5. Ускорение параллельного алгоритма

В дальнейшей работе планируется проведение более масштабных экспериментов.

Заключение

В статье были рассмотрены вопросы, связанные с переносом понятий и алгоритмов, разработанных в математической теории фракталов, на приложения реляционных баз данных. Исследованы известные приложения теории фракталов к обработке данных. Предложен новый алгоритм фрактального поиска в реляционной базе данных, позволяющий обнаруживать повторяющиеся группы данных. На основе предложенного алгоритма разработана программная система в контексте приложений СУБД Oracle 10g. Для наиболее трудоемкого этапа алгоритма выполнена параллельная реализация в рамках парадигмы MapReduce. Проведены вычислительные эксперименты, показавшие достаточно высокую эффективность предложенного решения.

В качестве направления дальнейшего развития работы можно рассмотреть разработку полностью параллельного алгоритма фрактального поиска.

Литература

1. Adibi, J. Fractals and Self-similarity in Data Mining: Issue and Approaches — KDD-2002 Workshop Report. — 2002. / J. Adibi, C. Faloutsos. URL: http://www.sigkdd.org/ sites/default/files/issues/4-2-2002-12/adibi.pdf (дата обращения: 1.08.2014).
2. Barbara D. Fractal Mining — Self Similarity-based Clustering and its Applications // Data Mining and Knowledge Discovery Handbook. — Springer. — 2010. — P. 573–589.

3. Ferreira Cordeiro, R.L. Clustering very large multi-dimensional datasets with MapReduce / R.L. Ferreira Cordeiro, C. Traina Jr, A.J. Traina, J. López, U. Kang, C. Faloutsos // Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining (KDD '11). — 2011. — P. 690–698.
4. Lämmel, R. Google's MapReduce programming model / R. Lämmel // Science of computer programming. — 2008. — 70(1). — P. 1–30.
5. Peres, S.M. A fractal fuzzy approach to clustering tendency analysis / S.M. Peres, M. L. de Andrade Netto // Advances in Artificial Intelligence–SBIA 2004. – Springer Berlin Heidelberg. — 2004. — P. 395–404.
6. Sousa, E. A fast and effective method to find correlations among attributes in databases / E. Sousa, C. Traina, A. Traina // Data Mining and Knowledge Discovery. — 2007. — 14(3). — P. 367–407.
7. Stonebraker, M. MapReduce and parallel DBMSs: friends or foes? / M. Stonebraker // Communications of the ACM. — 2010. — P. 64–71.
8. Traina Jr, C. Fast feature selection using fractal dimension/ C. Traina Jr, A. Traina, L. Wu, C. Faloutsos // Journal of Information and Data Management. — 2010. —Vol. 1, No. 1. — P. 3–16.
9. White, T. Hadoop: The definitive guide. / T. White — O'Reilly Media/Yahoo Press — 2012. — 688 p.
10. Yan, G. The practical method of fractal dimensionality reduction based on Z-ordering technique / G. Yan, Z. Li, L. Yuan // Advanced Data Mining and Applications. — 2006. — P. 542–549.
11. Zmeškal, O. Fractal analysis of image structures. / O. Zmeškal, M. Veselý, M. Nežádal, M. Buchníček // Harmonic and Fractal Image Analysis. — 2001. — P. 3–5.
12. Дейт, К.Дж. Введение в системы баз данных, 8-е издание. / К.Дж. Дейт — М.: Издательский дом «Вильямс», 2006. — 1328 с.
13. Кайт, Т. Oracle для профессионалов. Архитектура, методики программирования и основные особенности версий 9i, 10g и 11g. / Т. Кайт — М.: Издательский дом «Вильямс», 2013. — 848 с.
14. Лымарь, Т.Ю. Фрактальный поиск в базе данных с применением модели распределенных вычислений / Т.Ю. Лымарь, Т.С. Мантрова // Параллельные вычислительные технологии (ПаВТ'2014): Труды международной научной конференции (1–3 апреля 2014 г., г. Ростов-на-Дону). — Челябинск: Издательский центр ЮУрГУ, 2014. — С. 369.
15. Лымарь, Т.Ю. Параллельный алгоритм фрактального поиска в базе данных / Т.Ю. Лымарь, Н.Ю. Староверова // Параллельные вычислительные технологии (ПаВТ'2011): Труды международной научной конференции (Москва, 28 марта – 1 апреля 2011 г.). — Челябинск: Издательский центр ЮУрГУ, 2011. — С. 703.
16. Мандельброт, Б. Фрактальная геометрия природы. / Б. Мандельброт — Москва: Институт компьютерных исследований, 2002. — 656 с.
17. Официальный Портал Росреестра. URL: http://rosreestr.ru (дата обращения: 07.05.2014).
18. Официальный сайт ФГУП ФКЦ «Земля». URL: http://www.fccland.ru (дата обращения: 07.05.2014).

19. Уэлстид, С. Фракталы и вейвлеты для сжатия изображений в действии. / С. Уэлстид — М.: Издательство Триумф, 2003. — 320 с.

Лымарь Татьяна Юрьевна, к.ф.-м.н., доцент кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), lymarti@susu.ac.ru.

Мантрова Татьяна Сергеевна, магистрант, кафедра системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), tatiana.mantrova@gmail.com.

Староверова Наталья Юрьевна, программист, ООО «БТ-Челябинск» (Челябинск, Российская Федерация), snu1988@yandex.ru.

Поступила в редакцию 12 августа 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 61–74

DOI: 10.14529/cmse140404

FRACTAL SEARCH ALGORITHM IN RELATIONAL DATABASES

T.Yu. Lymar, South Ural State University (Chelyabinsk, Russian Federation),

T.S. Mantrova, South Ural State University (Chelyabinsk, Russian Federation),

N.Yu. Staroverova, ООО «BT-Chelyabinsk» (Chelyabinsk, Russian Federation)

The article deals with the development of algorithms of fractal analysis of relational databases. An overview and comparative analysis of the known applications of the theory of fractals in data processing is provided. A new algorithm of fractal search in a relational database, which allows detecting duplicate data group, is presented. Implementation of the proposed algorithm for the Oracle DBMS is considered. An implementation using distributed computing MapReduce paradigm is described. Examples of using the developed algorithm to compress and analyze the contents of the database are presented. The results of computational experiments are given.

Keywords: relational databases, the theory of fractals, fractal analysis of databases, data compression.

References

1. Adibi J., Faloutsos C. KDD-2002 Workshop Report. Fractals and Self-similarity in Data Mining: Issue and Approaches URL: http://www.sigkdd.org/sites/default/files/issues/4-2-2002-12/adibi.pdf (accessed: 1.08.2014).
2. Barbara D., Chen P. Fractal Mining — Self Similarity-based Clustering and its Applications // Data Mining and Knowledge Discovery Handbook. 2010. P. 573–589. DOI: 10.1007/978-0-387-09823-4_28.

3. Ferreira Cordeiro R.L., Traina Jr C., Traina A.J., López J., Kang U., Faloutsos C. Clustering very large multi-dimensional datasets with MapReduce // Proceedings of the 17th International Conference on Knowledge Discovery and Data Mining (KDD '11). 2011. P. 690–698. DOI: 10.1145/2020408.2020516.
4. Lämmel R. Google's MapReduce programming model // Science of computer programming. 2008. 70(1). P. 1–30.
5. Peres S.M., de Andrade Netto M.L. A fractal fuzzy approach to clustering tendency analysis //Advances in Artificial Intelligence — SBIA 2004. — Springer Berlin Heidelberg, 2004. P. 395–404. DOI: 10.1007/978-3-540-28645-5_40.
6. Sousa E.P., Traina Jr C., Traina A.J. A fast and effective method to find correlations among attributes in databases // Data Mining and Knowledge Discovery. 2007. 14(3). P. 367–407. DOI: 10.1007/s10618-006-0056-4.
7. Stonebraker M. MapReduce and parallel DBMSs: friends or foes? // Communications of the ACM, 2010. P. 64–71. DOI: 10.1145/1629175.1629197.
8. Traina Jr.C., Traina A., Wu L., Faloutsos C. Fast feature selection using fractal dimension // Journal of Information and Data Management. 2010. Vol. 1, No. 1. P. 3–16.
9. White T. Hadoop: The definitive guide. Yahoo Press, 2012. 688 p.
10. Yan G., Li Z., Yuan L. The practical method of fractal dimensionality reduction based on Z-ordering technique // Advanced Data Mining and Applications. 2006. P. 542–549. DOI: 10.1007/11811305_60.
11. Zmeškal O., Veselý M., Nežádal M., Buchníček M. Fractal analysis of image structures. // Harmonic and Fractal Image Analysis. 2001. P. 3–5.
12. Date C.J. An Introduction to Database System. Addison-Wesley, 2003. 1024 p.
13. Kyte T. Expert Oracle Database Architecture Oracle Database 9i, 10g, and 11g Programming Techniques and Solutions. Apress, 2010. 832 p.
14. Lymar T.Yu., Mantrova T.S. Fractalny poisk v base dannykh s primeneniem modeli raspredelennykh vychisleny [Fractal search the database using distributed computing]. Parallelnye vychislitelnye tekhnologii (PaVT'2014): Trudy mezhdunarodnoj nauchnoj konferentsii (Rostov-on-Don, 1–3 aprelya 2014) [Parallel Computational Technologies (PCT'2014): Proceedings of the International Scientific Conference (Rostov-on-Don, Russia, April, 1–3, 2014)]. Chelyabinsk, Publishing of the South Ural State University, 2014. P. 369.
15. Lymar T.Yu., Staroverova N.Yu. Parallelny algorithm fractalnogo poiska v base dannykh [Parallel algorithm of fractal database search]. Parallelnye vychislitelnye tekhnologii (PaVT'2011): Trudy mezhdunarodnoj nauchnoj konferentsii (Moscow, 28 marta – 1 aprelya 2011) [Parallel Computational Technologies (PCT'2011): Proceedings of the International Scientific Conference (Moscow, Russia, March, 28 – April, 1, 2011)]. Chelyabinsk, Publishing of the South Ural State University, 2011. P. 703.
16. Mandelbrot B. The Fractal Geometry of Nature. NY: W. H. Freeman and Company, 1982. 461 p.
17. Official Portal of Rosreestr. URL: https://rosreestr.ru (accessed: 07.05.2014).
18. Official site of the FCC FSUE «Zemlya». URL: http://www.fccland.ru (accessed: 1.02.2014).
19. Welstead S. Fractal and Wavelet Image Compression Techniques. SPIE Publications, 1999. 254 p.

Received August 12, 2014.

УДК 004.031.2, 004.382.2 DOI: 10.14529/cmse140405

КВАЗИПЛАНИРОВЩИК ДЛЯ ИСПОЛЬЗОВАНИЯ ПРОСТАИВАЮЩИХ ВЫЧИСЛИТЕЛЬНЫХ МОДУЛЕЙ МНОГОПРОЦЕССОРНОЙ ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЫ ПОД УПРАВЛЕНИЕМ СУППЗ[1]

А.В. Баранов, Е.А. Киселёв, Д.С. Ляховец

Рассматривается способ сокращения количества простаивающих ресурсов супер-ЭВМ за счет использования особенностей механизмов планирования заданий в современных системах пакетной обработки (СПО), в частности, особенностей алгоритма обратного заполнения (backfill algorithm). Рассмотрена схема работы разработанного авторами квазипланировщика — программного средства для утилизации простаивающих ресурсов супер-ЭВМ под управлением отечественной СПО — системы управления прохождением параллельных заданий (СУППЗ). Пользователи квазипланировщика при наличии простаивающих ресурсов могут без ожидания в общей очереди запускать на счет задания, выполнение которых возможно на произвольном числе процессоров из заданного пользователем диапазона. Приведены полученные в ходе недельного вычислительного эксперимента данные о влиянии работы квазипланировщика на показатели эффективности планирования — процент загрузки вычислителя и среднее приведенное время нахождения задания в очереди.

Ключевые слова: показатели эффективности планирования, утилизация ресурсов, квазипланировщик, алгоритм обратного заполнения

Введение

Многопроцессорная вычислительная система (ВС), как правило, состоит из управляющей ЭВМ и вычислителя (см. рис. 1). Вычислитель представляет собой совокупность вычислительных модулей (ВМ), объединенных коммуникационной средой.

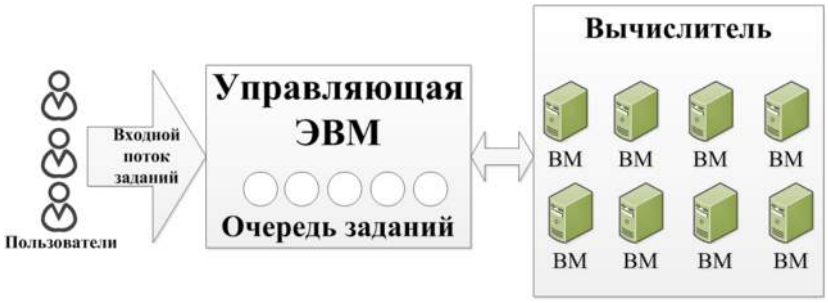

Рис. 1. Схема многопроцессорной вычислительной системы

Пользователи отправляют задания на выполнение, формируя входной поток заданий. Каждое отправленное задание должно быть снабжено файлом специального вида – паспортом, содержащим информацию о требуемом количестве ВМ для счета, планируемом времени выполнения и иных требованиях к ресурсам вычислителя. Поток заданий посту-

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

пает на управляющую ЭВМ, где формируется очередь заданий. Дождавшееся своей очереди задание поступает на выполнение на указанном пользователем количестве ВМ. Вышеперечисленные функции (прием входного потока пользовательских заданий, ведение очереди заданий и запуск заданий на счет) выполняет специальное программное обеспечение – система пакетной обработки (СПО), которая может быть размещена как только на управляющей ЭВМ, так и на управляющей ЭВМ и ВМ.

В отечественных многопроцессорных вычислительных системах много лет используется Система управления прохождением параллельных заданий (СУППЗ) — система планирования запуска и прохождения параллельных заданий, созданная в Институте прикладной математики им. М.В. Келдыша Российской академии наук (ИПМ им. Келдыша РАН) и Межведомственном суперкомпьютерном центре (МСЦ) РАН в 2001 году [1].

СУППЗ ведет очередь параллельных заданий, обеспечивая по возможности полную загрузку вычислителя. В СУППЗ планирование осуществляется на основе алгоритма обратного заполнения (т.н. backfill-планировщик). В СУППЗ алгоритм обратного заполнения был реализован впервые в мировой практике построения СПО для многопроцессорных ВС, на основе этого алгоритма С.В. Шарфом (ИММ УРО РАН г. Екатеринбург) был реализован планировщик — сервер очередей, являющийся ядром СУППЗ [2].

Алгоритм обратного заполнения позволяет использовать простаивающие процессоры для запуска заданий с низким приоритетом вне очереди, если их выполнение не повлияет на время старта более приоритетных заданий. Такое возможно, например, в случае, если для задания А, стоящего в очереди ранее, недостаточно ресурсов, и при этом задание Б, стоящее в очереди позже, успеет завершиться до момента, когда освободится достаточное количество ресурсов для запуска задания А. Никакое менее приоритетное задание не может занять процессоры так, чтобы это отодвинуло старт более приоритетного. Планировщик определяет время завершения выполняющихся заданий и освобождения занятых процессоров, используя заказанное время выполнения, указанное в паспорте задания.

Эту особенность алгоритма обратного заполнения можно использовать для повышения загрузки вычислителя. Для этого необходимо отслеживать простаивающие процессоры и помещать в очередь такие задания, которые поступят на выполнение без ожидания в очереди.

1. Показатели качества планирования

Обычно выделяют следующие показатели качества планирования:

1. Утилизация ресурсов (загрузка вычислителя).

2. Среднее значение времени нахождения задания в очереди относительно заказанного времени его счета.

Под утилизацией ресурсов будем понимать отношение задействованного при выполнении пользовательских заданий параллельного ресурса (совокупности ВМ) ко всему объему параллельного ресурса. Если в течение 5 часов работали 10 ВМ, и в течение каждого часа один процессор простаивал в ожидании задания (см. рис. 2), то утилизация составит 80 %.

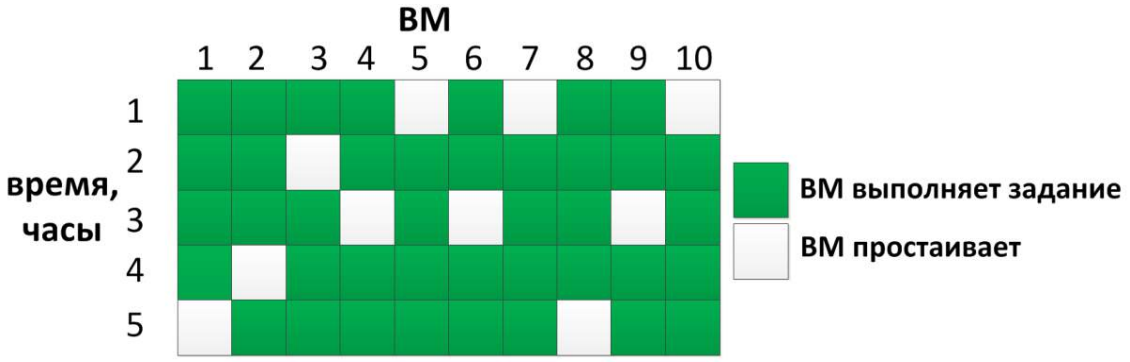

Рис. 2. Расчет утилизации ресурсов вычислителя

Согласно статистическим данным работы [3] утилизация ресурсов систем под управлением СУППЗ находится на уровне 95 %. Для больших ВС простаивание ресурсов около 5 % оказывается весьма значительным, поскольку это эквивалентно тому, что один из 20 ВМ не работает. Для системы в 2000 ВМ в среднем простаивает 100 ВМ.

Предположим, что Q_i — *время, прошедшее с момента попадания i-го задания в очередь до начала его выполнения,* R_i — *заказанное время выполнения i-го задания.*

Величину времени нахождения задания в очереди относительно заказанного времени его счета обозначим как T_i*:*

$$T_i = \frac{Q_i}{R_i}. \tag{1}$$

Тогда для k заданий можно определить величину T_{mid} – *среднее значение времени нахождения задания в очереди относительно заказанного времени счета:*

$$T_{mid} = \frac{\sum_{i=1}^{k} T_i}{k} = \frac{1}{k} \cdot \sum_{i=1}^{k} \frac{Q_i}{R_i}. \tag{2}$$

Приведение ко времени счета осуществляется потому, что для разных заданий одно и то же время ожидания может означать разное качество планирования. Например, время ожидания 30 минут будет вполне удовлетворительным для задания с заказанным временем счета 10 часов, но то же время ожидания (30 мин.) будет неприемлемым для задания со счетом в 5 минут. Очевидно, что чем меньше T_{mid}, тем выше качество планирования.

2. Повышение утилизации ресурсов за счет использования квазипланировщика

Современные высокопроизводительные ВС могут состоять из значительного (несколько сотен или тысяч) числа ВМ и обслуживать интенсивный входной поток пользовательских заданий. Некоторые ВМ будут неизбежно простаивать. Так, например, если следующему в очереди заданию B требуется 10 ВМ, а на текущий момент доступно лишь 9, то эти 9 ВМ будут простаивать, пока освободится еще 1 ВМ, после чего задание B запустится на требуемых ему 10 ВМ (см. рис. 3).

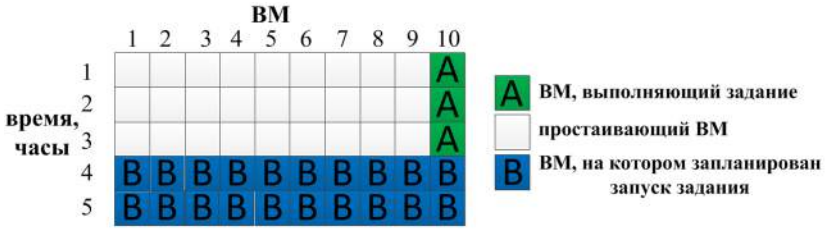

Рис. 3. Задание В ожидает запуска через 3 часа, 9 ВМ простаивают

Назовем совокупность простаивающих ВМ **окном** в плане запуска заданий. Под размером окна будем понимать количество простаивающих ВМ или процессоров и время их простоя. На рис. 3 приведено окно размером 9 ВМ на 3 часа (состоит из белых квадратов). Аналогично, **размером** задания назовем требуемое заданию число ВМ или процессоров и заказанное время счета задания. Если размеры задания не превышают размера окна, то задание может быть запущено в окне плана запуска заданий (т.е. на простаивающих ВМ) без ожидания в очереди. Запуск такого задания будем называть **заполнением окна** плана запуска заданий.

Авторами разработано программное средство (ПС) определения окон в плане запуска заданий планировщика СУППЗ, которое способно определять размеры окон текущего расписания. Размер окна представляет собой максимальный размер задания, которое будет запущено на счет без ожидания в очереди.

Если в очереди найдется задание С, размеры которого не превышают размеров окна, то алгоритм обратного заполнения позволит запустить задание С на счет. Запуск задания С не задержит старт более приоритетного задания Б. Например, низкоприоритетное задание С размером 1 ВМ на 2 часа будет запущено на счет в окне размером 9 ВМ на 3 часа без ожидания в очереди (см. рис. 4).

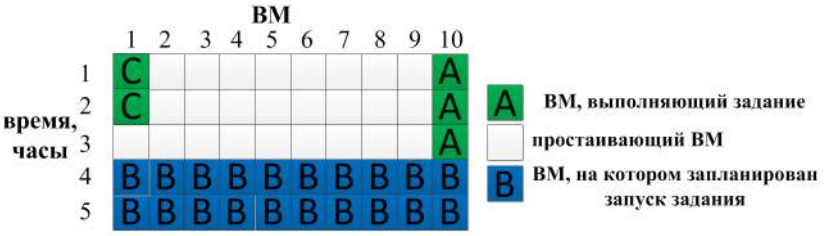

Рис. 4. Низкоприоритетное задание С запущено на 1 ВМ и 2 часа счета

Существует категория заданий, для которых требование к числу ВМ не является строгим. Такие задания предпочтительно запустить на 9 ВМ без ожидания, чем на 100 ВМ, но с длительным ожиданием. Одним из решений для таких пользователей может стать **квазипланировщик**, позволяющий указывать в паспорте нижнюю и верхнюю границы требуемого числа ВМ. Например, «9-100» должно означать, что задание предпочтительно запустить на 100 ВМ, однако возможно запустить и на любом количестве ВМ от 9 до 100, если это позволит уменьшить время пребывания в очереди. Квазипланировщик (см. рис. 5) определяет окна в плане запуска, принимает от пользователей задания и ставит в очередь планировщику СУППЗ задания, для которых достаточно простаивающих ресурсов. В силу особенности алгоритма обратного заполнения, СУППЗ будет запускать полученные от квазипланировщика задания на простаивающих ресурсах без ожидания в общей очереди.

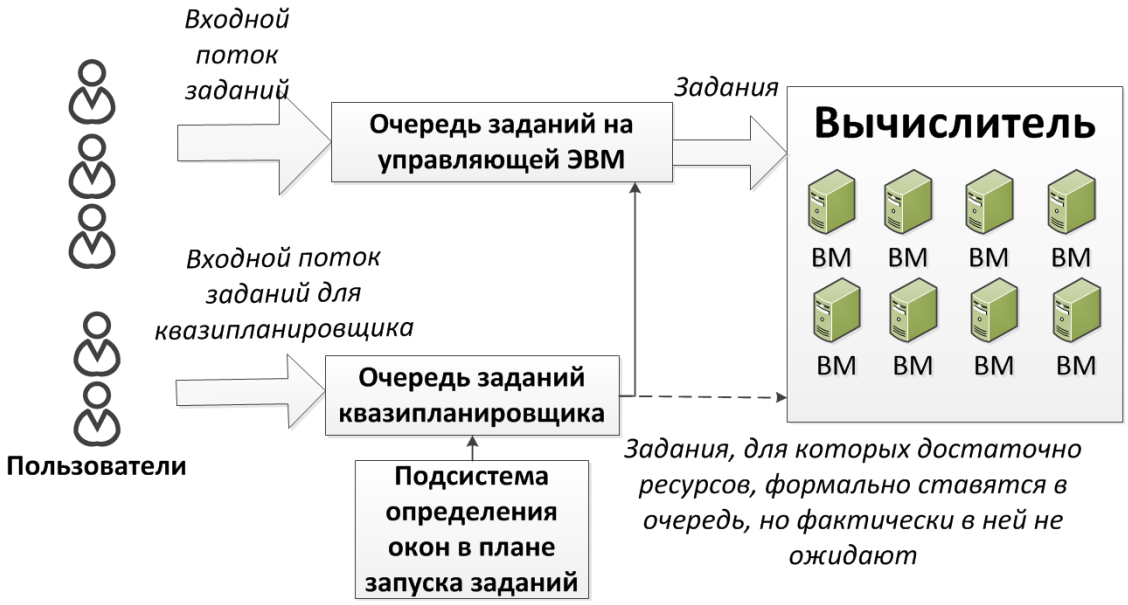

Рис. 5. Схема совместной работы СУППЗ и квазипланировщика

Термин «квазипланировщик» использован авторами по следующей причине. С точки зрения пользователя в системе появляется еще одна планируемая очередь для специального типа заданий. При этом реального планирования этой очереди не происходит, квазипланировщик лишь отслеживает окна в плане запуска общей очереди заданий и «подмешивает» в нее задания из своей очереди.

Попытки использования простаивающих ресурсов могут негативно отразиться на качестве обслуживания текущих заданий. В примере заполнения окна низкоприоритетным заданием (см. рис. 4) 1 ВМ из 9 свободных был задействован для расчета задания С из конца очереди. Планирование запуска заданий происходит из заказанного пользователем времени счета, а реальное время счета может быть существенно меньше заказанного. Согласно статистике работы СУППЗ *только у 10 % заданий реальное время счета совпадает с заказанным*. Свыше 60 % заданий завершаются за час и более до окончания заказанного времени счета. Задание А может завершиться раньше, чем через 3 часа, и ожидаемый 1 ВМ освободится раньше планируемого.

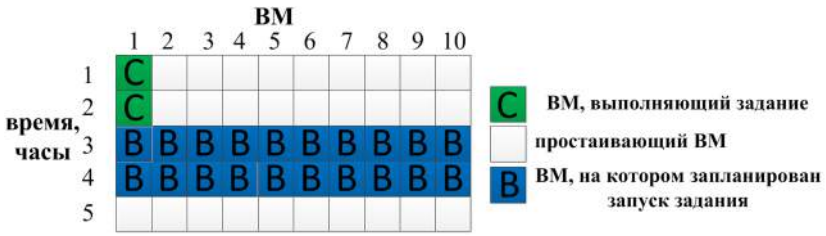

Рис. 6. Низкоприоритетное задание С мешает запуску приоритетного задания В

Разница между заказанным и реальным временем счета приводит к нежелательной ситуации (см. рис. 6) — менее приоритетное задание С, запущенное на 1 ВМ, мешает запуску более приоритетного В, и 9 ВМ будут ожидать завершения работы этого менее приоритетного задания в течение 2 часов. Подобные ситуации могут привести к тому, что качество обслуживания заданий в целом не улучшится, а ухудшится. Очевидно, что работа квазипланировщика будет оказывать влияние на показатели качества планирования

СУППЗ, причем характер этого влияния заранее предсказать невозможно. Поэтому необходимо провести экспериментальное исследование влияния квазипланировщика на качество планирования.

Проведение подобных исследований на реальной ВС в реальном масштабе времени сопряжено с большими расходами вычислительных мощностей и времени, поскольку расчет показателей качества эффективности планирования имеет смысл проводить за большие промежутки времени (больше нескольких дней). Это приводит к необходимости создания и использования симулятора СУППЗ, который позволит сократить время проведения исследования за счет продвижения модельного времени.

Квазипланировщик одинаково функционирует на симуляторе и на реальной системе. Если использование квазипланировщика на симуляторе приведет к положительным изменениям показателей эффективности, то можно будет задействовать квазипланировщик на реальной системе.

3. Свойства разработанного симулятора СУППЗ

Режим симуляции не предполагает реального выполнения заданий, поэтому на вход симулятора должен поступать автоматическим образом сформированный входной модельный поток заданий. Во время своей работы СУППЗ сохраняет параметры запущенных заданий в базе данных (БД) «Статистика» (см. рис. 7). В частности, в этой БД сохраняются время поступления задания в очередь, требуемое число процессоров и заказанное время счета. Для создания входного модельного потока заданий из БД «Статистика» реальной СУППЗ можно извлечь параметры выполненных заданий.

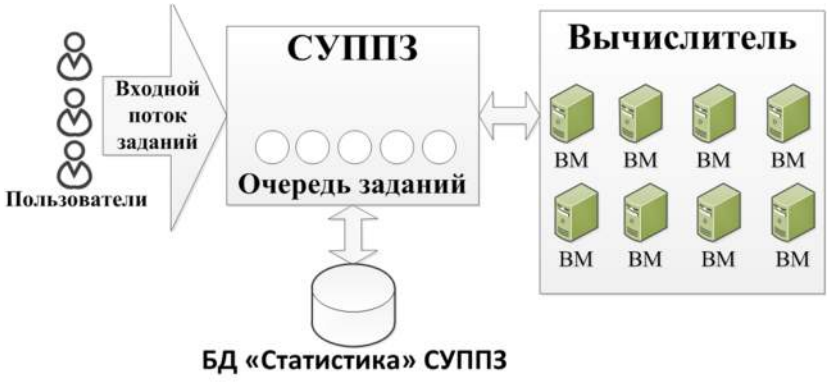

Рис. 7. Схема работы СУППЗ

Для исследования влияния работы квазипланировщика на показатели качества планирования СУППЗ необходимо запустить симулятор без квазипланировщика и с квазипланировщиком. Поиск оптимальных параметров квазипланировщика потребует многократного запуска симулятора с различными параметрами.

Перед каждым запуском симулятора СУППЗ должна иметь одинаковое начальное состояние. Для получения начального состояния СУППЗ возможно сохранить файлы конфигурации, расписания, состояния планировщика и очереди (вместе с паспортами стоящих в очереди заданий) реальной СУППЗ.

Восстановив начальное состояние СУППЗ, необходимо изменить исследуемые настройки, влияющие на характеристики системы. К таким настройкам будем относить параметры квазипланировщика и множество входных заданий для него.

Свойства симулятора СУППЗ:

1. Симулятор функционирует без реального многопроцессорного вычислителя. Для этого задания запускают фиктивно, без реального выполнения.

2. Модельный поток заданий, поступающий на вход симулятора СУППЗ, имеет те же статистические характеристики, что и реальный поток заданий. Для этого авторами была создана и задействована специальная подсистема формирования модельного потока заданий. Эта подсистема извлекает из БД «Статистика» параметры выполненных в реальной СУППЗ заданий и преобразует их во входной поток заданий для симулятора СУППЗ.

3. Запущенные на фиктивный счет задания завершаются не по окончании заказанного времени счета, а по окончании реального времени счета из статистики работы реальной СУППЗ.

4. Для ускорения проведения симуляции возможно осуществлять продвижение модельного времени.

4. Схема работы квазипланировщика

Как уже отмечалось, существует категория заданий, для которых требование к числу ВМ не является строгим. Задания, выполнение которых возможно на произвольном числе процессоров из заданного диапазона, образуют входной поток заданий для квазипланировщика (см. рис. 8). Из поступающих заданий квазипланировщик формирует собственную очередь.

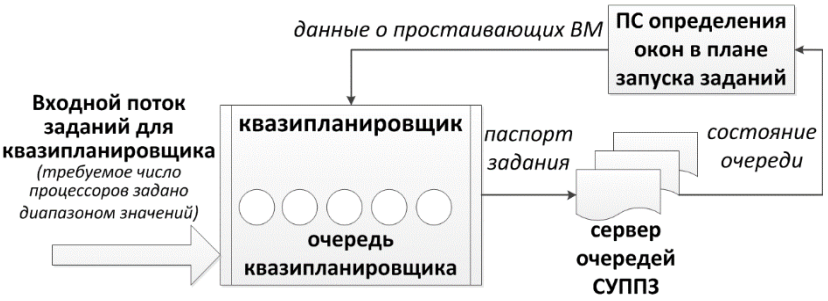

Рис. 8. Схема работы квазипланировщика

Подсистема определения окон в плане запуска заданий анализирует текущее состояние очереди и сообщает квазипланировщику размер ближайшего окна. Квазипланировщик рассматривает очередное задание в очереди и определяет возможность его запуска в ближайшем окне. Если запуск возможен, квазипланировщик формирует и ставит в очередь СУППЗ задание. Благодаря особенности алгоритма обратного заполнения формально поставленные в очередь СУППЗ задания фактически в ней не ожидают. Если запуск следующего задания невозможен, то квазипланировщик будет ожидать информацию о вновь появившихся окнах в плане запуска.

Если пользователь поставил в очередь квазипланировщика 10-минутное задание с требуемым числом процессоров в диапазоне от 9 до 100, и, согласно текущему плану запуска заданий, простаивает 50 процессоров в течение 15 минут, то возможно немедленно запустить на счет задание на 50 процессоров на 10 минут.

5. Влияние квазипланировщика на показатели эффективности СУППЗ

Для проведения вычислительного эксперимента использовался журнал работы отечественной супер-ЭВМ МВС-100К (МСЦ РАН) за неделю с 10:30 19.03.2014 по 10:00 26.03.2014. Начальное состояние СУППЗ было получено в 10:30 19.03.2014 во время штатной профилактики, когда никаких пользовательских заданий на вычислителе не выполняется. В очереди на момент начала профилактики находилось 42 задания.

В журнале работы за недельный период были найдены все задания, которые при завершении не освободили занятые процессоры. Такие процессоры СУППЗ автоматически блокирует и исключает из состава вычислителя. Соответствующие задания в модельном потоке были снабжены дополнительным параметром – числом неосвобожденных при завершении процессоров.

За указанный период было запущено и завершено 4088 заданий согласно журналу работы реальной СУППЗ и 803 задания было запущено квазипланировщиком. Входной поток заданий для квазипланировщика состоял из однотипных заданий разных пользователей. Заказанное время счета всех заданий 20 минут. Заказанное число процессоров указано диапазоном от 8 до 32. Реальное время счета 20 минут.

Симуляция недели модельного времени заняла около трех суток. Рассчитанные показатели утилизации для симулятора без квазипланировщика и с квазипланировщиком приведены в таблице.

Таблица

Влияние квазипланировщика на утилизацию ресурсов МВС-100К

Дата	**Утилизация без квазипланировщика, %**	**Утилизация с квазипланировщиком, %**	**Выгода %**
19.03.14	88,1	88,8	0,7
20.03.14	95,1	97,0	1,9
21.03.14	93,4	94,0	0,6
22.03.14	95,3	95,0	-0,3
23.03.14	93,1	94,9	1,8
24.03.14	96,3	96,9	0,6
25.03.14	91,9	93,0	1,1
Среднее арифметическое	**93,31**	**94,23**	

В большинстве случаев квазипланировщик оказывает положительное влияние на среднюю утилизацию ресурсов вычислителя. Отрицательное влияние имеет место быть из-за разницы между заказанным временем счета и реальным временем счета заданий основного входного потока СУППЗ.

Среднее значение времени нахождение задания в очереди относительно заказанного времени его счета составило 0,62 для симулятора без квазипланировщика и 0,54 для симулятора с квазипланировщиком. При этом среднее время ожидания для заданий основ-

ного входного потока незначительно возросло и составило 0,63. Для поставленных в очередь квазипланировщиком заданий среднее время ожидания в очереди находилось на уровне 0,001.

Заключение

Авторами впервые была поставлена и решена задача создания квазипланировщика для утилизации простаивающих ресурсов системы под управлением СУППЗ. В ходе вычислительного эксперимента на симуляторе СУППЗ пользователям квазипланировщика предоставлялась возможность без ожидания в очереди запускать на счет задания, выполнение которых возможно на произвольном числе процессоров из заданного диапазона. Запуск таких заданий не оказал существенного отрицательного влияния на качество обслуживания текущих пользователей СУППЗ и позволил незначительно повысить среднюю утилизацию ресурсов.

Разработанный авторами квазипланировщик может быть внедрен в работу реальной СУППЗ на многопроцессорной вычислительной системе для повышения утилизации ресурсов и улучшения качества обслуживания пользователей квазипланировщика при отсутствии значительного влияния на среднее время ожидания в очереди заданий входного потока СУППЗ.

Литература

1. Система управления прохождением параллельных заданий [Электронный ресурс]. URL: http://suppz.jscc.ru (дата обращения: 13.07.2014).
2. Шарф, С.В. Обслуживание очереди задач и многофакторные приоритеты // Параллельные вычисления в ИММ УрО РАН [Электронный ресурс]. URL: http://parallel.imm.uran.ru/pubs/izhevsk03-scharf.htm (дата обращения: 13.07.2014).
3. Баранов, А.В. Сравнение систем пакетной обработки с точки зрения организации промышленного счета / А.В. Баранов, А.В. Киселев, В.В. Старичков, Р.П. Ионин, Д.С. Ляховец // Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17–22 сентября 2012 г., г. Новороссийск). М.: Изд-во МГУ, 2012. С. 506.

Баранов Антон Викторович, старший научный сотрудник, Межведомственный суперкомпьютерный центр РАН (Москва, Российская Федерация), antbar@mail.ru

Киселёв Евгений Андреевич, сотрудник, Межведомственный суперкомпьютерный центр РАН (Москва, Российская Федерация), kiselev@jscc.ru

Ляховец Дмитрий Сергеевич, сотрудник, Межведомственный суперкомпьютерный центр РАН (Москва, Российская Федерация), anetto@inbox.ru

Поступила в редакцию 20 августа 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 75–84

DOI: 10.14529/cmse140405

THE QUASI SCHEDULER FOR UTILIZATION OF MULTIPROCESSING COMPUTING SYSTEM'S IDLE RESOURCES UNDER CONTROL OF THE MANAGEMENT SYSTEM OF THE PARALLEL JOBS

A.V. Baranov, Joint Supercomputer Center of Russian Academy of Science (Moscow, Russia), antbar@mail.ru

E.A. Kiselev, Joint Supercomputer Center of Russian Academy of Science (Moscow, Russia), kiselev@jscc.ru

D.S. Lyakhovets, Joint Supercomputer Center of Russian Academy of Science (Moscow, Russia), anetto@inbox.ru

The paper considers an approach to reduce an amount of multiprocessing computing system's idle resources. The presented approach uses features of job scheduling mechanisms in modern batch systems, in particular, features of backfill algorithm. Authors developed the quasi scheduler, program that utilizes supercomputer idle resources under parallel job management system control (SUPPZ). Flowchart of the developed quasi scheduler is presented. Quasi scheduler's users can submit tasks without waiting in queue, if there are enough idle resources for task execution. Computational experiment's results presented in paper show effectiveness of supercomputer's scheduling depends of the quasi scheduler usage.

Keywords: effectiveness of scheduling, resource utilization, extra scheduler, backfilling algorithm.

References

1. Sistema upravleniya prohozhdeniem parallelnih zadaniy [Parallel job management system]. URL: http://suppz.jscc.ru (accessed: 13.07.2014).
2. Scarf S.V. Obsluzhivanie ocheredi zadach i mnogofactornie prioritety [Handling jobs queue and multifactor priority] // Parallelnie vichisleniya v IMM UrO RAN [Parallel calculation in Institute of Mathematics and Mechanics, Ural Branch of the Russian Academy of Science]. URL: http://parallel.imm.uran.ru/pubs/izhevsk03-scharf.htm (accessed: 13.07.2014).
3. Baranov A.V., Kiselev A.V., Starichkov V.V., Ionin R.P., Lyakhovets D.S. Sravnenie system paketnoy obrabotki s tochki zreniya organizacii promishlennogo scheta [Control batch systems comparison in terms of industrial calculations] // Nauchniy servis v seti Internet: poisk novih resheniy: Trudi Mezhdunarodnoy superkomputernoy konferencii (Novorossiysk, 17–22 sentyabrya 2012) [Science service in the Internet: Searching the new solutions: Proceeding of the International Supercomputer Conference (Novorossiysk, Russia, 17–22, September, 2012)]. Moscow, Publishing of the Moscow State University, 2012, 506 p.

Received August 20, 2014.

УДК 004.75 DOI: 10.14529/cmse140406

ОРГАНИЗАЦИЯ ДОСТУПА К ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛИТЕЛЬНЫМ РЕСУРСАМ В HPC COMMUNITY CLOUD[1]

С.А. Вайцель, М.А. Городничев

HPC Community Cloud представляет собой программный комплекс для объединения вычислительных ресурсов в единый сервис и предоставления доступа к этому сервису пользователям через веб-приложение и внешним программным системам через программный интерфейс. HPC Community Cloud скрывает нюансы работы с различными вычислительными системами за единой точкой доступа пользователей к сервису. В работе представлена архитектура и реализация программного комплекса HPC Community Cloud.

Ключевые слова: суперкомпьютеры, распределенные вычисления, облачный сервис, высокопроизводительные вычисления.

Введение

Цель проекта HPC Community Cloud (HPC2C) [1] — создание программного инструментария для обеспечения доступа программных систем к высокопроизводительным вычислительным системам (ВВС), накопления и повторного использования программ для решения прикладных задач на ВВС, повышения уровня интерфейсов взаимодействия пользователей с ВВС.

Программный инструментарий HPC2C состоит из веб-приложения, реализующего пользовательский интерфейс, и сервера управления, предоставляющего программный интерфейс внешним программным системам. Высокоуровневый веб-интерфейс HPC2C для взаимодействия пользователей с высокопроизводительными вычислительными системами предназначен для повышения эффективности работы пользователей научно-образовательных вычислительных центров, снижает порог вхождения пользователей в область высокопроизводительных вычислений за счет интерактивных обучающих материалов и интуитивно понятных инструментов управления. Сервер управления ведет учет пользователей, подключенных высокопроизводительных вычислительных систем, управляет прохождением расчетных задач на вычислительных системах, организует хранение пользовательских данных и программ. Программный интерфейс обеспечивает возможность разработки прикладных программных систем, способных обращаться к ресурсам вычислительных центров для проведения крупномасштабных расчетов. HPC2C скрывает нюансы работы с различными вычислительными системами за единой точкой доступа пользователей к сервису и единой системой управления.

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

HPC2C реализует платформу для накопления и интегрирования в единый интерфейс содержимого, создаваемого сторонними разработчиками: средств конструирования программ, средств визуализации данных, интерактивных учебных материалов, средств численного моделирования, анализа данных.

В настоящее время пользователям вычислительных систем, как правило, приходится при переходе от системы к системе изучать нюансы взаимодействия с ними. Перенос задач из одной в систему в другую не автоматизирован, пользователь обычно все делает вручную. Отсутствует интеграция инструментов интерактивного обучения работе с высокопроизводительными вычислительными системами и программирования таких систем со средствами доступа к высокопроизводительным вычислительным системам.

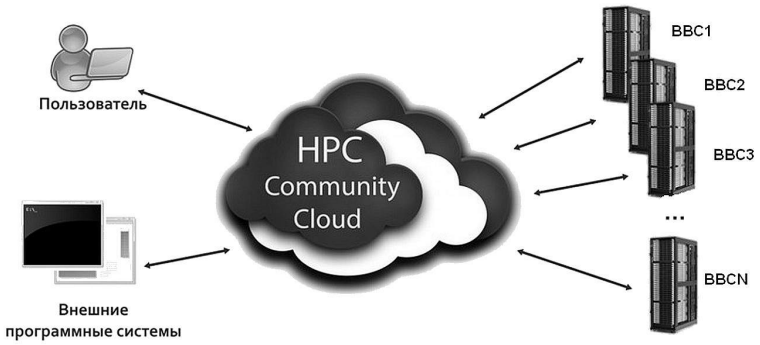

Рис. 1. HPC2C — единый интерфейс для доступа к множеству вычислительных систем

Коммерческие центры обработки данных предоставляют облачные сервисы в области высокопроизводительных вычислений (Microsoft Azure Big Compute, Amazon EC2 и др.). Однако в настоящее время сервис сводится к тому, что пользователь может сконструировать для себя нужных размеров и свойств виртуальный суперкомпьютер, а далее он встречается с теми же самыми проблемами программирования и взаимодействия с вычислительными системами, что и в случае с традиционными вычислительными центрами. Компания Sabalcore создала облачный сервис, который предоставляет доступ к формируемому самой компанией набору программных пакетов для численного моделирования, при этом отсутствует возможность для пользователей и сторонних разработчиков свободно публиковать свои программные пакеты, учебные материалы.

В разделе 1 представлена архитектура программного комплекса HPC2C и программный интерфейс сервера управления HPC2C. В разделе 2 описано веб-приложение HPC2C. В разделе 3 даны подходы к интеграции пользовательских приложений в среду HPC2C.

1. Архитектура программного комплекса HPC Community Cloud

Программный инструментарий HPC Community Cloud состоит из веб-приложения и сервера управления, предоставляющего программный интерфейс внешним программным системам (рис. 2). Посредством веб-интерфейса пользователи HPC Community Cloud разрабатывают программы, осуществляют работу с ранее установленными в HPC Community Cloud программами численного моделирования, анализа данных, ставят вычислительные задания на исполнение на ВВС. Сервер управления ведет учет пользователей, подключенных ВВС, управляет прохождением расчетных задач на вычислительных системах, организует хранение пользовательских данных и программ. Функциональность сервера

доступна внешним программным системам (примером которых является веб-приложение) посредством программного интерфейса (API) HPC Community Cloud. Программный интерфейс обеспечивает возможность разработки прикладных программных систем, способных обращаться к ресурсам вычислительных центров для проведения крупномасштабных расчетов.

Рис. 2. Архитектура программного инструментария HPC2C

1.1. Функциональность сервера управления

К большинству функций сервера управления могут обращаться внешние программные системы посредством программного интерфейса. Программный интерфейс реализован в виде HTTP-сервиса, принимающего запросы определенного вида. Для облегчения разработки внешних приложений, работающих на стороне браузера, а также приложений на платформе Node.js, создана библиотека функций на JavaScript, реализующих соответствующие запросы.

В ведении сервера управления находятся среди прочего следующие объекты:

- пользователи,
- подключенные вычислительные системы,
- вычислительные задания, которые ставят пользователи на вычислительные системы,
- приложения,
- файлы,
- базы данных.

Неформально функционирование HPC Community Cloud описывается следующим образом. Пользователь загружает в HPC Community Cloud файлы с исходными данными и текстами программ, регистрирует доступные ему вычислительные системы, на которых он имеет собственные учетные записи, либо же выбирает вычислительные системы их списка общедоступных подключенных в HPC Community Cloud систем, описывает правила сборки программ на основе системы make, осуществляет сборку программ в соответствии с архитектурой целевой вычислительной системы, ставит вычислительное задание на исполнение, отслеживает состояние задания, имеет доступ к файлам, сформированным в ходе расчетов. Для формирования заданий пользователи могут также выбирать приложения, к которым им предоставили доступ другие пользователи, а также предоставлять доступ к своим приложениям. Все указанные функции доступны через программный интерфейс сервера управления, а пользователи получают доступ к этим функциям либо

посредством веб-приложения HPC2C, либо внешних программных систем. В HPC2C вводятся и используются понятия фреймворков (каркасов) программ и моделей. Фреймворки — это предметно-ориентированные средства разработки программ, обладающие встроенным в веб-приложение HPC2C графическим интерфейсом, который проводит пользователя через серию шагов по дополнению фреймворка до исполняемой программы. Модели — это готовые программы численного моделирования с графическим интерфейсом, в котором пользователь задает параметры расчетов и отправляет сформированное вычислительное задание на выполнение на какой-либо вычислительной системе. Модели могут встраиваться в веб-приложение HPC2C, если для реализации их взаимодействия с пользователем не требуются функции за пределами API HPC Community Cloud. Для некоторых моделей может оказаться целесообразным реализация отдельного веб-приложения, серверная часть которого проводит обработку данных либо генерацию элементов пользовательского интерфейса, которые не могут быть эффективно реализованы на клиентской стороне (в браузере на JavaScript) с обращениями к API HPC Community Cloud.

В дальнейших разделах рассматривается спецификация объектов управления порядка работы с ними.

1.2. Учет и авторизация пользователей

Обращение ко всем функциям сервера управления осуществляется веб-приложением или внешними программными системами от имени зарегистрированных пользователей посредством программного интерфейса.

Регистрация пользователей осуществляется через форму в веб-приложении или непосредственно через программный интерфейс.

Работа с программным интерфейсом начинается с авторизации пользователя по паре <логин, пароль> и получению, в случае успешной авторизации, ключа доступа — строки специального вида. В дальнейших обращениях к функциям программного интерфейса наряду с остальными параметрами посылается ключ доступа, который служит подтверждением права вызывающей стороны на выполнение этой функции. Через определенный период времени ключ доступа дезактивируется и должен быть запрошен новый.

Пользователи могут состоять в различных группах. Права на выполнение различных функций над различными объектами могут выставляться на уровне групп и на уровне отдельных пользователей

Сервер ведет базу данных пользователей. Программный интерфейс HPC Community Cloud включает следующие операции для управления учетными записями:

- создание учетной записи с заданными полями,
- изменение полей учетной записи
- проверка пары <логин, пароль>, т.е. авторизация, и предоставление ключа доступа к остальным функциям.

1.3. Операции с файлами

Сервер управления осуществляет управление хранением пользовательских файлов, перемещение файлов между вычислительными системами перед выполнением и после выполнения расчетных заданий, позволяет выгружать файлы на машину пользователя и загружать файлы с машины пользователя в HPC Community Cloud, редактировать структуру файлового дерева.

При создании учетной записи пользователя для этой учетной записи создается в файловом хранилище «домашний» каталог со следующей структурой:

/apps
/appstorage
/experiments
/frameworks
/models

Каталог /apps предназначен для хранения файлов приложений, разрабатываемых через интегрированную среду разработки (IDE), встроенную в веб-приложение (см. раздел 2).

В каталоге /appstorage размещают свои рабочие данные различные приложения, фреймворки и модели. Каталог играет такую же роль, как папка Application Data в профилях пользователей Windows.

В каталоге /experiments для каждого задания, которое ставится на исполнение на вычислительные системы, создается каталог с именем задания. В каталоге размещаются исполнительные файлы, входные и выходные файлы расчетов, прочие необходимые для выполнения задания материалы.

В каталоге /frameworks размещаются каталоги фреймворков. Фреймворки могут быть разработаны пользователем, могут быть загружены извне. Список фреймворков, доступных пользователю, состоит из фреймворков в этом каталоге и фреймворков, к которым предоставили ему доступ другие пользователи.

В каталоге /models размещаются каталоги моделей. Модели могут быть разработаны пользователем через IDE, могут быть загружены извне. Список моделей, доступных пользователю состоит из моделей в этом каталоге и моделей, к которым предоставили ему доступ другие пользователи

Программный интерфейс HPC Community Cloud включает следующие операции для управления файлами:

- загрузка файла в хранилище HPC Community Cloud,
- выгрузка файла из хранилища,
- получение списка файлов и подкаталогов в заданном каталоге,
- копирование файлов/каталогов между расположениями в хранилище,
- перемещение файлов/каталогов между расположениями в хранилище (в т.ч. переименование),
- создание каталогов,
- удаление файлов/каталогов из хранилища.

Программный интерфейс не позволяет пользователям обращаться к файлам вне их домашнего каталога.

1.4. Учет подключенных вычислительных систем

Пользователи подключают к своей учетной записи вычислительные системы, после чего могут осуществлять запуск заданий на этих вычислительных системах через программный интерфейс или веб-приложение.

Можно использовать вычислительные системы, находящиеся в списке общедоступных в HPC Community Cloud или подключать к HPC Community Cloud новые вычислительные системы, в том числе персональные рабочие станции с известными IP-адресами. Можно сделать такие системы доступными для подключения другими пользователями.

Для подключения новой системы нужно указать следующие данные:

- имя вычислительной системы,
- доменное имя или IP-адрес,
- имя учетной записи пользователя на этой вычислительной системе,
- пароль или ключ доступа к вычислительной системе.

Персональные пароль или ключ доступа сохраняются в базе данных HPC Community Cloud.

1.5. Выполнение расчетных задач

Сервер управления HPC2C от имени пользователя, используя его логин и пароль/ключ, осуществляет

- запуск заданий на подключенных к аккаунту пользователя вычислительных системах,
- мониторинг состояния заданий,
- загрузку необходимых для выполнения задания файлов на вычислительную систему из файлового хранилища HPC Community Cloud,
- загрузку файлов, полученных в результате расчетов, из вычислительной системы в файловое хранилище HPC Community Cloud,
- управление ходом вычислений на основании команд пользователя, подаваемых в интерактивном режиме.

Программный интерфейс предоставляет единый формат для описания задания для всех подключаемых систем. В зависимости от особенностей порядка прохождения задач на той или иной системе, описание задание транслируется в необходимый формат, и выполняются необходимые действия для запуска задания. Для обеспечения возможности запуска задач на системах, порядок прохождения задач в которых отличается от тех, что уже зарегистрированы в HPC Community Cloud, может потребоваться либо реализация отдельного модуля для взаимодействия с данной вычислительной системой, либо могут быть расширены или переконфигурированы существующие модули. В настоящее время поддерживается запуск последовательных и параллельных, реализованных с применением технологии MPI, программ на рабочих станциях и кластерах на Linux без системы управления прохождением задач (очереди), и системой прохождения задач типа OpenPBS (Torque). В зависимости от специфических настроек системы управления прохождением задач, выполняемых администраторами, может потребоваться настройка модулей HPC Community Cloud, ответственных за взаимодействие с такими системами, например, подстановка конкретного имени очереди.

Сведения, необходимые для формирования задания в HPC Community Cloud обычны для систем управления прохождением задач в высокопроизводительных вычислительных системах и в грид.

Постановка задания на счет состоит из двух этапов, для чего предусмотрены две разные команды в программном интерфейсе: формирование задания и запуск задания. Все файлы, необходимые для выполнения задания и получаемые в результате выполнения задания создаются в каталоге /experiments/имя_задания. Сформированное задание может быть модифицировано: все поля, за исключением названия задания. Задание может быть запущено многократно.

1.6. Разработка пользовательских приложений и сборка

Для обеспечения разработки пользовательских приложений и их сборки предусмотрены такие операции:

- создание приложения (создается каталог приложения в директории /apps и осуществляется регистрация приложения в базе данных),
- модификация дерева файлов приложения (создание каталогов, файлов, и т.д.),
- спецификация порядка сборки приложения через систему сборки make,
- сборка приложения.

В настоящее время поддерживается сборка последовательных приложений на C/C++ и параллельных на C/C++ и LuNA [2] с использованием MPI или NumGRID [3].

2. Веб-приложение HPC Community Cloud

Веб-приложение реализует пользовательский интерфейс к системе HPC2C. Посредством этого интерфейса пользователь осуществляет разработку приложений, постановку вычислительных заданий на счет, отслеживание состояния заданий и просмотр результатов расчетов. Веб-приложение HPC2C является примером внешней программной системы, эксплуатирующей программный интерфейс HPC2C для доступа к системе хранения данных и управления прохождением заданий в высокопроизводительных вычислительных системах. Веб-приложение изолирует пользователя от особенностей различных вычислительных систем, с которыми он теперь работает через единый интерфейс. Веб-приложение состоит из трех основных инструментов: личного кабинета (рис. 3), среды разработки программ (IDE, рис. 4), интерфейса для постановки вычислительных заданий (рис. 5). В личном кабинете пользователь видит список всех доступных приложений и может создать новое приложение, видит статус всех созданных вычислительных заданий и может сформировать новое или модифицировать и перезапустить старые.

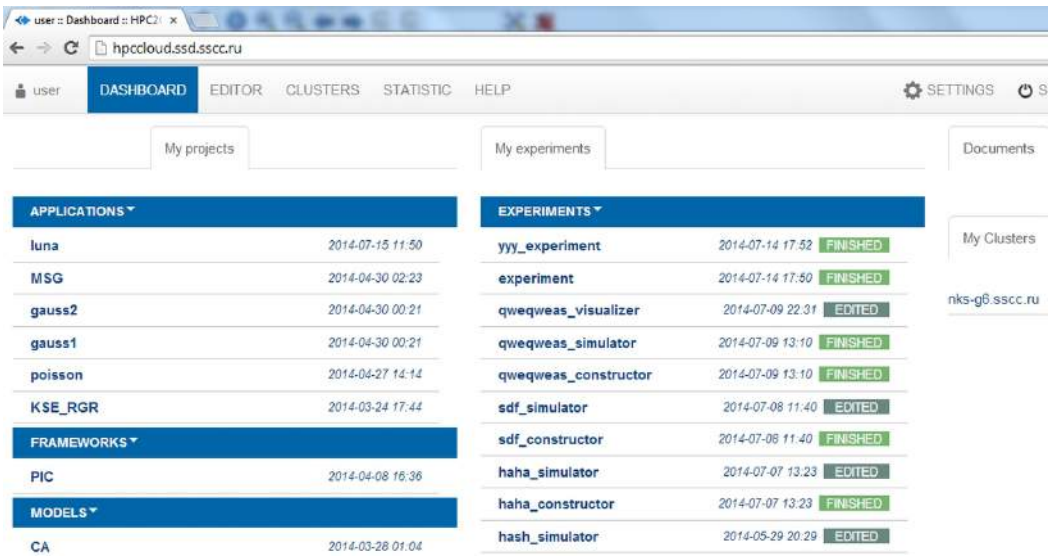

Рис. 3. Личный кабинет пользователя HPC2C

Среда разработки программ имеет типичную для таких систем функциональность, но доступна через Интернет в браузере пользователя, и непосредственно в ней может быть сформировано задание на исполнение разработанной программы на подключенных к ак-

каунту пользователя высокопроизводительных вычислительных системах. Окно формирования задания позволяет задать параметры вычислительного задания, обычные для высокопроизводительных вычислительных систем. В личном кабинете пользователю доступен список фреймворков и моделей, на основе которых он может сформировать задания (см. раздел 3).

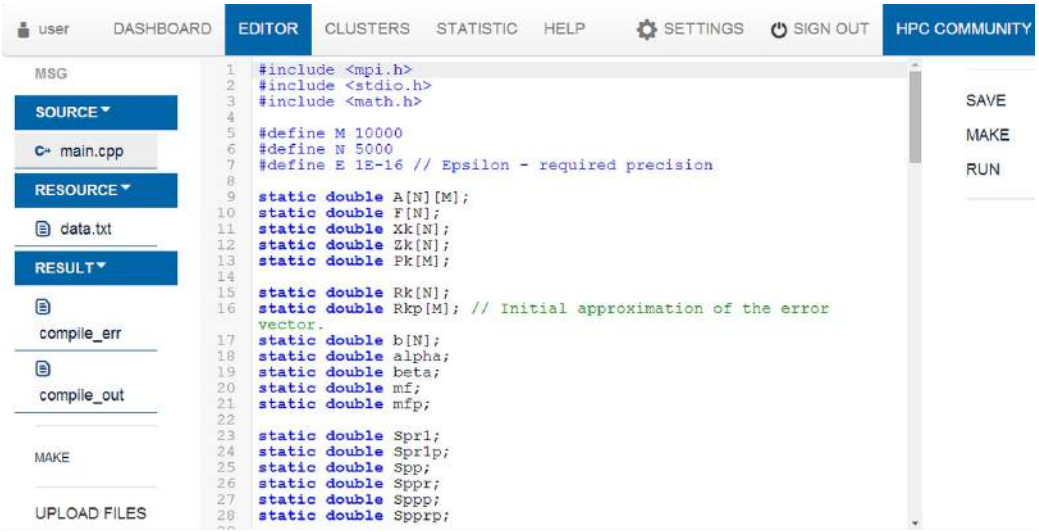

Рис. 4. IDE для онлайн-разработки

3. Встраивание пользовательских моделей и фреймворков в веб-приложение HPC Community Cloud

Организация веб-приложения HPC2C предполагает возможность интеграции в него интерфейсных средств для моделей и фреймворков, разрабатываемых пользователями. Все необходимые файлы для работы моделей и фреймворков размещаются пользователем в файловом хранилище HPC2C в специальных каталогах /models и /frameworks. По нажатии пользователем на имя модели или фреймворка в списке в личном кабинете веб-приложение загружает подготовленный разработчиком интерфейса и правильным образом размещенный файл, содержащий код на JavaScript, который реализует пользовательский интерфейс модели (фреймворка). Код содержит реализацию инициализирующей функции с заданным прототипом, которую вызывает веб-приложение после загрузки скрипта. Для работы с файловым хранилищем HPC2C, постановки расчетных задач, мониторинга расчетных задач и выгрузки результатов работы для демонстрации пользователю скрипт использует вызовы к JavaScript библиотеке HPC2C, реализующей обращения к API HPC2C.

На основе этого подхода разработан встроенный в веб-приложение HPC2C интерфейс для программного комплекса клеточно-автоматного моделирования газопорошковых потоков Ю.Г. Медведева (ИВМиМГ) [4]. Интерфейс обеспечивает задание параметров моделирования, в т.ч. модельной области с различными вариантами стенок, препятствий в каналах, и обеспечивает визуальное представление результатов расчетов. Вычисления выполняются на кластере Сибирского суперкомпьютерного центра. Пользователь в своем личном кабинете может отслеживать состояние задания, возвращаться к работе с моделью, анализировать результаты и запускать повторные вычисления.

Интерактивные обучающие материалы реализуются и встраиваются в веб-приложение HPC2C на основе этой технологии.

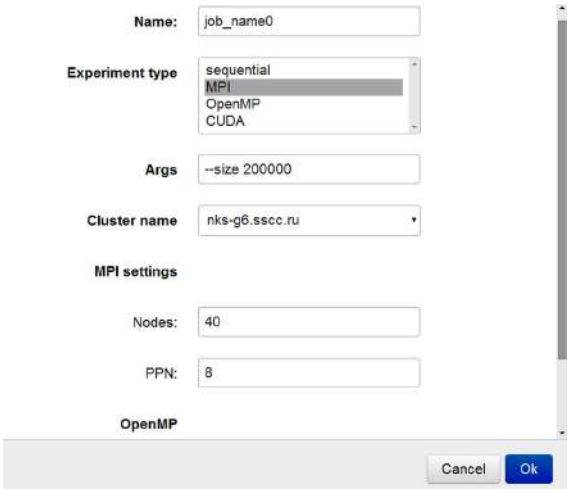

Рис. 5. Интерфейс для постановки расчетных заданий

Заключение

Разработана архитектура и реализован прототип программного комплекса HPC Community Cloud в составе сервера, обеспечивающего учет пользователей, вычислительных ресурсов, управление хранением данных и прохождением заданий в вычислительных системах и предоставляющего услуги внешним программным системам через программный интерфейс, а также веб-приложения, являющегося примером внешней программной системы, и реализующего высокоуровневый пользовательский интерфейс для удаленной работы с высокопроизводительными вычислительными ресурсами. Архитектура системы позволяет встраивание прикладных программных систем, интерактивных обучающих материалов в единый интерфейс взаимодействия с высокопроизводительными вычислительными ресурсами.

Литература

1. Городничев, М.А. HPC Community cloud: эффективная организация работы научно-образовательных суперкомпьютерных центров / М.А. Городничев, В.Э. Малышкин, Ю.Г. Медведев // Научный вестник НГТУ. — 2013. — № 3(52). — С. 91–96.
2. Malyshkin, V. Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system / V. Malyshkin, V. Perepelkin // The Journal of Supercomputing, Special issue on Enabling Technologies for Programming Extreme Scale Systems. — 2012. — Vol. 61, No. 1. — P. 91–96. — DOI: 10.1007/s11227-011-0649-6.
3. Городничев, М.А. Объединение вычислительных кластеров для крупномасштабного численного моделирования в проекте NumGRID / М.А. Городничев // Вестник Новосибирского государственного университета (серия «Информационные технологии»). — 2012. — Т. 10, вып. 4. — С. 63–73.
4. Медведев, Ю.Г Программный комплекс клеточно-автоматного моделирования газопорошковых потоков / Ю.Г. Медведев // Параллельные вычислительные технологии (ПаВТ'2012): труды международной научной конференции (Новосибирск, 26–30 марта 2012 г.). — Челябинск: Издательский центр ЮУрГУ, 2012. — С. 732.

Городничев Максим Александрович, младший научный сотрудник Института вычислительной математики и математическо геофизики СО РАН (Новосибирск, Российская Федерация), maxim@ssd.sscc.ru.

Вайцель Сергей Александрович, студент, Новосибирский государственный технический университет (Новосибирск, Российская Федерация), seralwei@gmail.com

Поступила в редакцию 12 августа 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 85–95

DOI: 10.14529/cmse140406

ORGANIZATION OF ACCESS TO SUPERCOMPUTING RESOURCES IN THE HPC COMMUNITY CLOUD

M. Gorodnichev, Institute of Computational Mathematics and Mathematical Geophysics SB RAS (Novosibirsk, Russian Federation),

S. Vaycel, Novosibirsk State Technical University (Novosibirsk, Russian Federation)

HPC Community Cloud is a software system for aggregation of computing resources into a single entry service and provisioning of access to these resources for users through web-application and for external software systems through application programming interface. HPC Community Cloud encapsulates peculiarities of access to different computing systems, allows users to accumulate and share content such as applications, data and documents. The paper presents the architecture and prototype implementation of the HPC Community Cloud.

Keywords: supercomputers, HPC, high performance computing, cloud service.

References

1. Gorodnichev M.A., Malyshkin V.E., Medvedev Yu.G. HPC Community cloud: effectivnaya organizatziya raboty nauchno-obrazovatel'nykh supercomputernykh tzenrov [HPC Community Cloud: Efficient Organization of Work at Scientific-Educational Supercomputing Centers] // Nauchny vestnik NGTU [Scientific Bulletin of NSTU], Novosibirsk, Publishing of Novosibirsk State Technical University. No. 3(52). 2013. P. 91–96.
2. Malyshkin V., Perepelkin V. Optimization methods of parallel execution of numerical programs in the LuNA fragmented programming system // The Journal of Supercomputing, Special issue on Enabling Technologies for Programming Extreme Scale Systems. Vol. 61, No. 1. 2012. P. 235–248. DOI: 10.1007/s11227-011-0649-6.
3. Gorodnichev M.A. Obyedineniye vychislitel'nykh klasterov dlya krupnomasshtabnogo chislennogo modelinrovania v proyekte NumGRID [Aggregation of Computing Clusters for Large-Scale Numerical Simulation in the NumGRID Project] // Vestnik Novosibirskogo Gosudarstvennogo Universiteta (seriya Informatzionnyje Tekhnologii) [Bulletin of

Novosibirsk State University (Information Technologies series)], Novosibirsk, Publishing of Novosibirsk State University. Vol. 10, Issue 4. 2012. P. 63–73.

4. Medvedev Yu.G. Programmny kompleks kletochno-avtomatnogo modelirovaniya gasoporoshkovykh potokov [Software system for cellular-automata simulation of gas-powder flows] // Parallelnye vychislitelnye tekhnologii (PaVT'2012): Trudy mezhdunarodnoj nauchnoj konferentsii (Novosibirsk, 26–30 marta 2012) [Parallel Computational Technologies (PCT'2010): Proceedings of the International Scientific Conference (Novosibirsk, Russia, March 26–30, 2012]. Chelyabinsk: Publishing of the South Ural State University, 2012. P. 732.

Received August 12, 2014.

DOI: 10.14529/cmse140407

INVESTIGATION OF DIFFERENT TOPOLOGIES OF NEURAL NETWORKS FOR DATA ASSIMILATION

F.P. Härter, H.F. Campos Velho

Neural networks have emerged as a novel scheme for a data assimilation process. Neural network techniques are applied for data assimilation in the Lorenz chaotic system. A radial basis function and a multilayer perceptron neural networks are trained employing 1000, 2000, and 4000 examples. Three different observation intervals are used: 0.01, 0.06 and 0.1 s. The performance of the data assimilation technique is investigated for different architectures of these neural networks.

Keywords: data assimilation, Neural Network, Data Assimilation.

Introduction

Data assimilation is a very important process in the numerical weather forecast. It permits the imbedding of observational data in the meteorological model. This data provides a feedback during the generation of the forecast in a real time fashion. However, the process of imbedding the observational data is not straightforward and it has to be done in a very smooth manner in order to minimize the propagation of errors in the forecast model. Usually, the assimilation process can be outlined as a two step iterative process:

Forecast step: $w_n^f = F[w_{n-1}^a]$

Analysis step: $w_n^a = w_n^f + d_n$

where w_n represents model state variable at time step n; $F[.]$ is the mathematical (forecast) model, superscripts f and a denote forecast and analyzed values respectively, and d_n is the innovation of the observational data. Several methods of data assimilation have been developed for air quality problems [1], numerical weather prediction [2], and numerical oceanic simulation [3]. In the case of atmospheric continuous data assimilation there are many deterministic and probabilistic methods. Deterministic approaches include dynamic relaxation, variational methods and Laplace transform, whereas probabilistic approaches include optimal interpolation and Kalman Filtering. In the Kalman filtering, the analysis innovation d_n is computed as a linear function of the misfit between observation (superscript o) and forecast (superscript f):

$$d_n = G_n(w_n^o - H_n w_n^f) \tag{1}$$

where G_n is the weight (gain) matrix, w_n^o is the observed value of w_n and H_n is the observation matrix. The Kalman filter has been tested in strongly nonlinear dynamical systems for assimilation procedure, such as the Lorenz chaotic system. Kalman filtering has the advantage of minimizing the error in the assimilation *plus* propagating this minimized error from one data insertion to the next. However, this process involves a heavy computational load, in particular for large meteorological systems. A strategy to alleviate this load is the use of artificial neural networks (ANN) to emulate the accuracy of the Kalman filtering [4]. Neural networks can be efficiently applied to map two data sets [5]. Several

architectures have been proposed for neural networks. The current work is based on the application of neural networks with *backpropagation* learning for data assimilation.

This paper deals with two neural networks: radial basis function and multilayer perceptron. These ANNs are employed for data assimilation for the Lorenz chaotic system [6]. Three different sizes of training set are used: 1000, 2000, and 4000 examples (patterns). Some numerical experiments are carried out for each training set, considering several time-periods for inserting the observations: 0.01, 0.06, and 0.1 seconds. The quality in the assimilation process is analyzed relating to the number of neurons, and different activation functions in the out-put layer. ANNs with two hidden layers are also studied in a class of experiments.

The next section provides a brief introduction to the neural network architecture used for the data assimilation application, and an outline on Kalman filter is presented too. However, it is not the aim of this paper to present an overview of ANNs. A further section discusses some numerical results. The final section adds some comments and remarks.

1. Non-linear model and assimilation processes

The framework used to perform the numerical experiments for the data assimilation is introduced.

A – The Lorenz Model

The Lorenz system [6] is a hard test for data assimilation, due to the fact it can present a chaotic dynamics. The equations for the Lorenz system are given by

$$\frac{dX}{dt} = -\sigma(X - Y), \tag{2}$$

$$\frac{dY}{dt} = RX - Y - XZ, \tag{3}$$

$$\frac{dZ}{dt} = XY - bZ. \tag{4}$$

This system is integrated using the predictor-corrector method with $\Delta t = 0.001$, using the following initial conditions (the subscript 0 denotes the initial condition): $X_0 = 1.508870$, $Y_0 = -1.531271$, $Z_0 = 25.460910$. The parameters in the system are: $\sigma = 10$, $b = 8/3$, and $R = 28$, so that the system is in the chaotic state.

B – Artificial Neural Networks

ANNs are mathematical models useful for carrying out some learning tasks, such as pattern recognition, function approximation, control, and filtering [5]. Figure 1 displays an outline of an ANN.

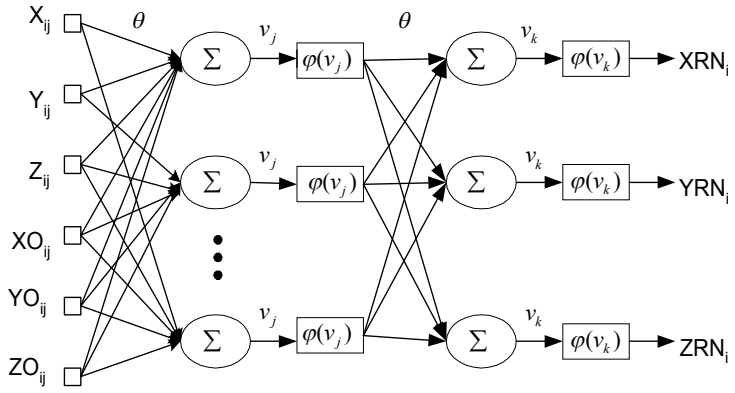

Fig.1. Sketch for neural networks used in this paper

The application of an ANN is done at two phases: learning and activation. The learning phase (also calling *training*) consists to find out the connection synaptic weights and bias associated with each neuron. Two strategies are possible for learning: supervised, and unsupervised. The main difference between supervised and unsupervised learning is that the latter uses only information contained in the input data, whereas the former requires both input and output (desired) data, which allows the calculation of the network error as the difference between the calculated output and the desired vector. In this paper the the supervised backpropagation learning process (Widrow's delta rule) [5] is used.

The activation is the process to obtain an output from an input for a given final architecture of the ANN. The activation function depends on the ANN topology used, for example:

$$\varphi(v_j) = \tanh(\frac{av_j}{2}) \quad (\text{with } a = 1) \tag{5}$$

is employed in the multilayer perceptron, and

$$\varphi(v_j) = \exp\left[-\frac{(v_j - \mu)^2}{2\sigma^2}\right] \quad (\sigma = 1 \text{ and } \mu = 0) \tag{6}$$

is used for radial basis funcions.

Different activation functions can be used for the out-put layer. Functions as given by equations (5) and (6) are tested, as well as linear function: $\varphi(v_j) = v_j$.

Multilayer percentron (MP)

The multilayer perceptron with backpropagation learning, or backpropagation neural network, is a feed-forward network composed of an input layer, an output layer, and a number of hidden layers for extracting high order statistics from the input data. Each of these layers may contain one or more neurons.

Mathematically, a perceptron network simply maps input vectors of real values into output vectors of real values. The connections in figure 1 have associated weights that are adjusted during learning process, thus changing the performance of the network. Neurons in the MP-NN are fully connected.

Radial Basis Functions (RBF)

Girosi and Poggio (1990) [7], based on Kolmogorov's theorem, show that ANNs with only one hidden layer are able to approximate any continuous function. The Girosi and Poggio's proof follows the Idea: a continuous and limited function can be consider as a combination of a linear Gaussians. These Gaussians can be implemented in the hidden layer. The accuracy of the approximation will depend on the number of Gaussian functions, i.e., the number of the neurons in the hidden layer.

ANNs representing functions fitted around a region, whose activation functions, implemented in the neurons of the hidden layer, are Gaussian ones, are examples of the radial basis functions neural network. For this ANN, learning means to find a surface in a multidimensional space, the best fit for the training data, where the agreement is measured in a statistical sense [5].

C – Kalman Filter (KF)

The KF is usefully used in estimation and control problems. Since its first applications on aerospace field [8], this technique has been employed in many applications. Recently, the KF has been applied to meteorology, oceanography and hydrology [2]. A brief description of the Kalman filter will be outlined here. Figure 2 shows an algorithm of the linear KF.

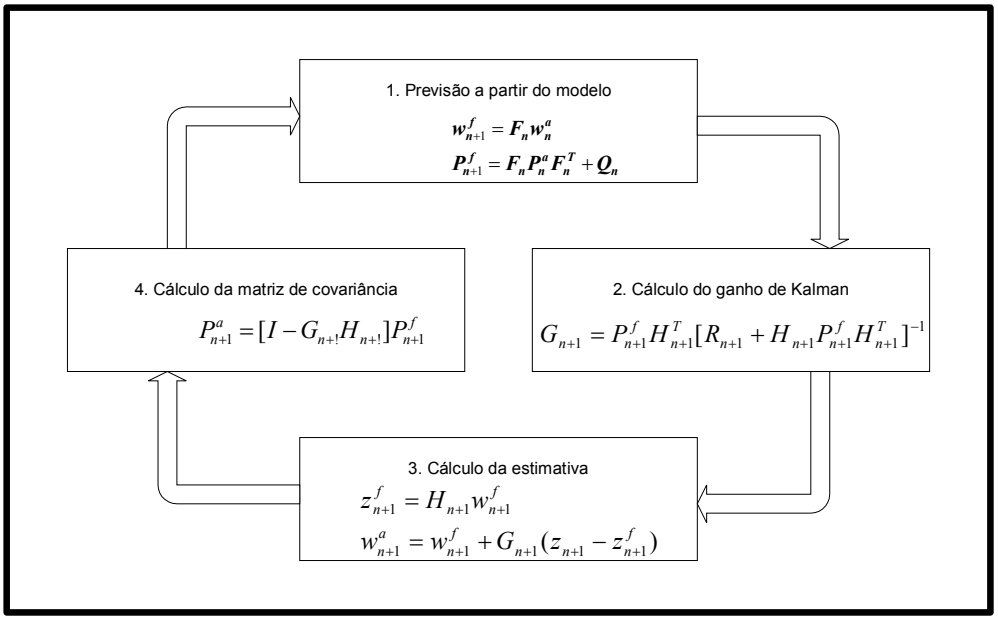

Fig 2. An outline of the Kalman filter algorithm

Let the prediction model be as in equation (7), where the subscript n denotes time-steps.

$$w_{n+1} = F_n w_n + \mu_n \qquad (7)$$

being F_n a mathematical description of the system, and μ_n a stochastic forcing (called dynamic modeling noise), and the observation model

$$z_n = H_n w_n + \upsilon_n \qquad (8)$$

where υ_n is a noise, and H_n represents the observation system. The typical gaussianity, zero-mean and ortogonality hypotheses for the noises are adopted. The term w_{n+1} is estimated through the recursion

$$w_{n+1}^a = (I - G_{n+1}H_{n+1})F_n c_n^a + G_{n+1}z_{n+1} \qquad (9)$$

where w_{n+1}^a is the estimator and G_n is the matrix that minimizes the trace of the prediction error covariance matrix, that is, the sum of the squares of the prediction errors in each component of W_{n+1}

$$J_{n+1} = E\{(w_{n+1}^a - w_{n+1})^T (w_{n+1}^a - w_{n+1})\} \qquad (10)$$

The algorithm of the KF is shown in figure 2, where Q_n is the covariance of μ_n, P_n^f is the covariance of the prediction errors, R_n is the covariance of υ_n, and P_n^a is the covariance of the estimation error. The assimilation is done from the sampled.

$$r(t_n + \Delta t) = r_{n+1} \equiv z_{n+1} - z_{n+1}^f = z_{n+1} - H_n w_{n+1}^f . \qquad (11)$$

2. Results and discussion

As mentioned before, the goal of this paper is to investigate the assimilation system based on ANN with different architectures (MP and RBF). Following this purpose, 396 experiments are performed.

For generating the training sets, the Lorenz system is integrated for 150000 time-steps (0.15 s), sampled at each 30 (0.003 s) producing 5000 examples. The first 4000 examples are applied in the training phase of the ANNs, and the rest of 1000 examples are used for the activation phase. The use of inputs that do not belong to the training set characterizes the *generalization capacity* of the ANN.

Following figure-1, the ANN inputs are normalized matrices $w = w(X, Y, Z)$ of the Lorenz system and $z = z(X_0, Y_0, Z_0)$ is the observation matrix. The desired output is the normalized matrix: $w_a = w_a(X_{FK}, Y_{FK}, Z_{FK})$, resulting from the assimilation with KF. The observations are synthetic ones, adding a Gaussian white noise, with variance 2, to the fields computed from the Lorenz system.

In the back-propagation algorithm, the synaptic weights are initialized according to the Gaussian distribution, and the training patterns are presented in the sequence as generated by the numerical model. ANNs were trained with learning ratio constant and equal to 0.1,

without momentum constant. One difference between MP-ANN and RBF-ANN is in the activation functions of the hidden layer: hyperbolic-tangent – equation (5) – for the former NN, and Gaussian function for the latter one – equation (6).

Figure 3 shows the relevance of the observation system. If there is no assimilation scheme, the disagreement between the computed dynamics (green curve) and true dynamics (observations – blue curve) becomes greater and greater.

Considering the large numbers of experiments, few results are shown. However, comments about our simulations are done.

The ANNs are trained with 1000, 2000, and 4000 examples, with data insertion (assimilation) performed at different time period: 0.01, 0.06, and 0.1 s. The number of neurons in the ANN varies from 3 up to 40, for both ANN. The activation function implemented for the hidden layer of the MP-NN is the hyperbolic-tangent for all experiments, while in the output layer the activation function is linear in the experiments 1 to 11 (C1 set), and hyperbolic-tangent in the experiments 12 to 22 (C2 set). For the RBF-NN the Gaussian function was implemented as activation function in the hidden layer for all experiments, while in the output layer the activation function is linear in the experiments 23 to 33 (C3 set), and Gaussian function in the experiments 34 to 44 (C4 set). Activation functions used here are summarized in the Table I.

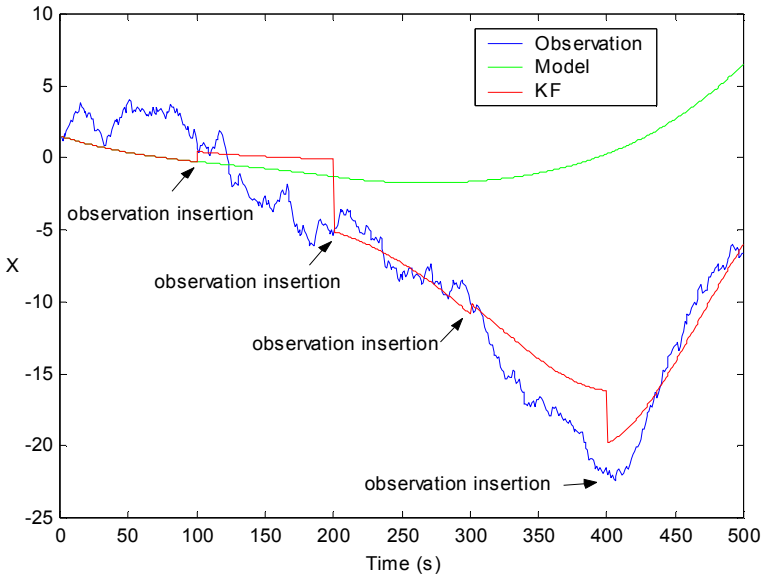

Fig. 3. Importance of the assimilation process

Table 1

Summary of experiments

Experiment	ANN	Output function
EXP1 to EXP11-C1	MP	linear
EXP12 to EXP22-C2	MP	tanh
EXP23 to EXP33-C3	RBF	linear
EXP34 to EXP44-C4	RBF	Gaussian

The quality of an assimilation system can be measured by the quadratic error, after the activation phase. This quantity is computed by the following equation:

$$RMS = \frac{1}{1000}\sum_{i=1}^{1000}(w_a - z)^2 \tag{12}$$

It was observed that when the error reached the value 0.0002 the ANN did not improve the solution. In fact, in many cases the output of the ANN with an error less than 0.0002 degraded the output. Therefore, the error equal 0.002 was defined as the target for training phase. Sometimes, the ANN did not reach this target.

Using 1000 patterns in the training set, with observation sampled at each 0.01 s, both ANNs with one hidden layer produce good results. For some architectures, the assimilation is better than that obtained with KF, whose the error 5.5764. The error for the best result using ANN for the C1 training set is 6.4610 (5 neurons), for o C2 training set is 4.6468 (3 neurons), for C3 training set is 4.5547 (8 neurons), and for C4 training set is 4.7280 (40 neurons). The MP-NN is defined having a linear activation function in the output layer. However, our experiments use hyperbolic-tangent in the output layer, the results are similar or even better when linear function is employed.

Figures 4-5 display the best results for MP-NN with 3 neurons (hyperbolic-tangent as the activation function in the output layer), and RBF-NN with 8 neurons (linear function in the output layer).

The experiments EXP37 (6 neurons) and EXP43 (30 neurons) do not show convergence. Figures 4-5 display the best results for MP-NN with 3 neurons (hyperbolic-tangent as the activation function in the output layer), and RBF-NN with 8 neurons (linear function in the output layer), respectively EXP12 and EXP28 experiments.

Figures 4 and 5 show assimilation results using ANNs (black line) and KF (blue line). Both procedures follow the dynamics of the system. However, one can not see from the figures which ANN produces the best result (smaller **RMS**). Computing **RMS** with equation (12) , the best result is obtained for the MP-NN (3 neurons), with **RMS** a little bit smaller than the best result for the RBF-NN. Experiments also show that ANNs with linear activation function in the output layer present worse results when hyperbolic-tangent and Gaussian functions are used as activation functions in the output layer for the MP-NN and RBF-NN, respectively.

For assimilation with sampled observation 0.06 s the estimative error with is 6.2377, i.e., increasing the time-period of observation the estimative is degraded. However, this conclusion does not apply to the assimilation for some architecture of ANNs. For example, in the C1 training set with sampled observation at 0.06 s, the experiment with 30 neurons presents **RMS** a thin smaller than same experiment sampled observation equal 0.01 s. The same occurs for experiments using C2 training set.

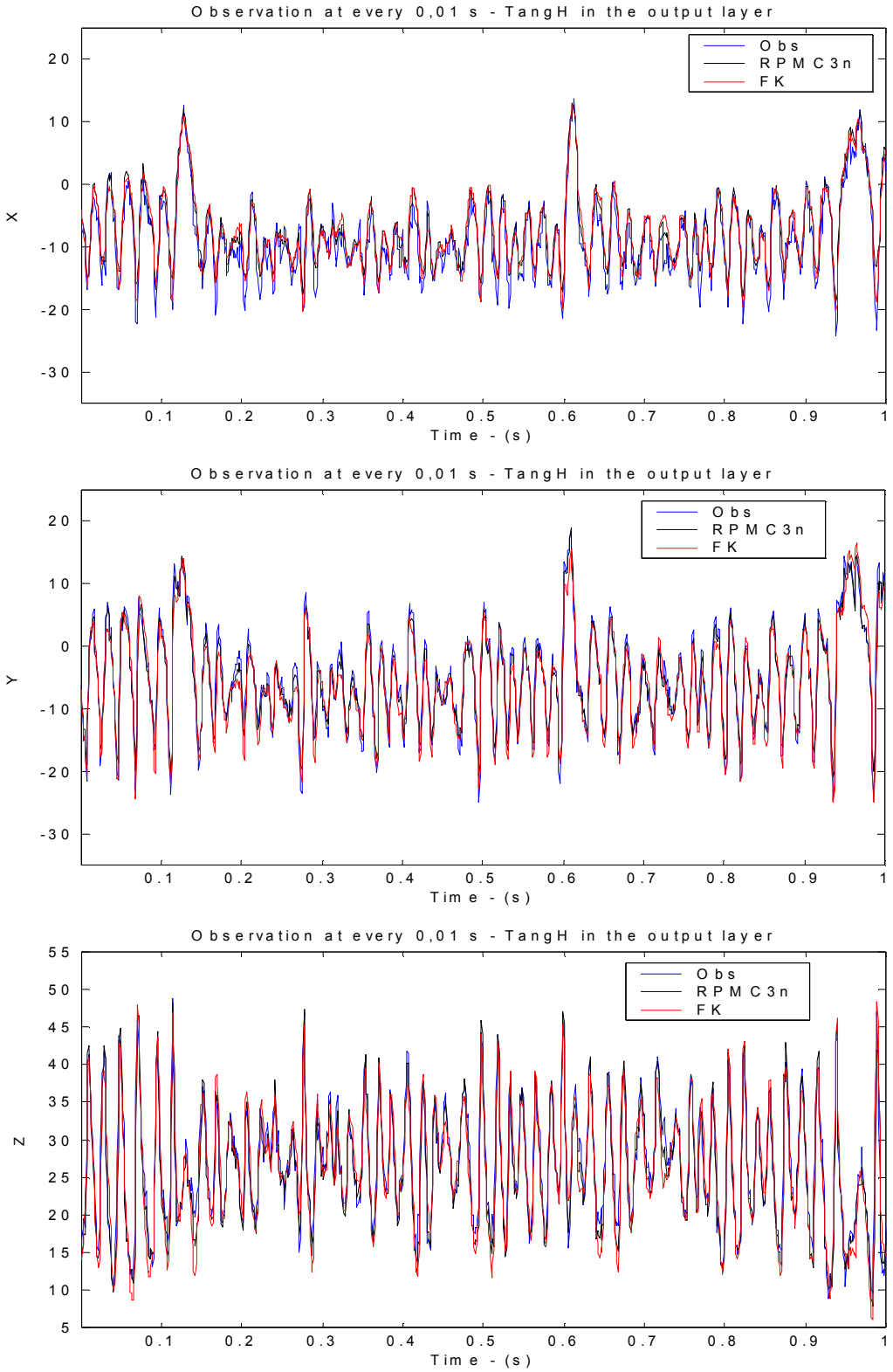

Fig. 4. Assimilation with MP-NN using 3 neurons. Hyperbolic-tangent was used in the output layer

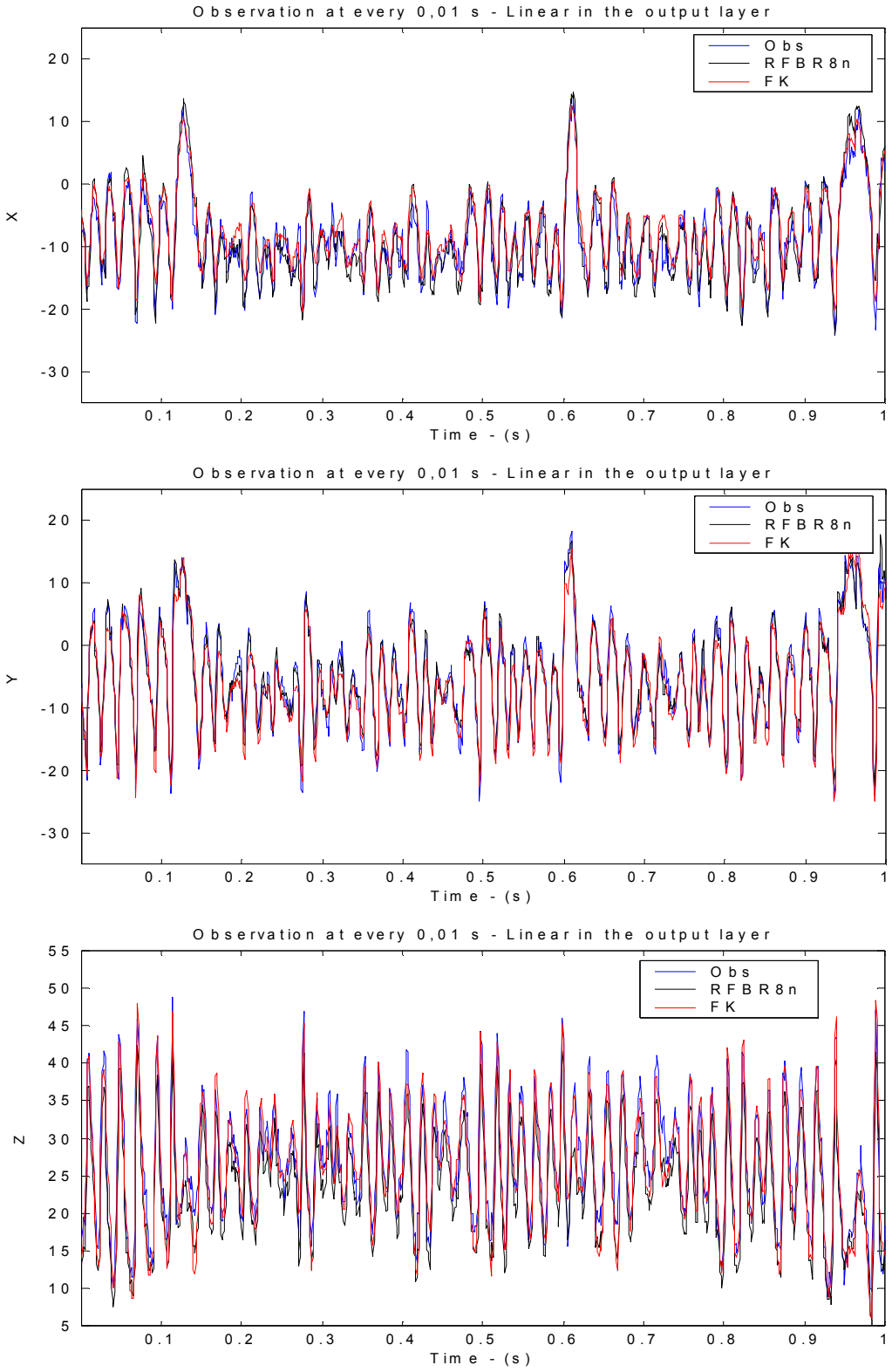

Fig. 5. Assimilation with RBF-NN using 8 neurons. Linear function was used in the output layer

The best results of the MP-NN for sampled observation at 0.06 and 0.01 s were obtained using 3 neurons, with hyperbolic-tangent in the output layer. For RBF-NN, the best

architecture was obtained with linear function in the output layer using 7 neurons and 8 neurons for sampled observation at 0.06 and 0,01 s, respectively.

The analysis of experiments with observations at each 0.1 s, the best architecture was obtained using 7 neurons in the hidden layer, with hyperbolic-tangent in the output layer. For RBF-NN, the best arrangement was gotten using 7 neurons, with linear function in the output layer.

Assimilation observations at each 0.1 s, the error of the estimative by KF increased, related to the experiments in which the assimilation was done at each 0.01 and 0.06 s, showing a **RMS**=9,4537. For the ANNs, the assimilation at each 0.1 s was degraded related to the experiments with observations inserted at 0.01 s, but this tendency is not verified for the most experiments in the C1 and C3 training sets.

It is hard to identify some pattern from the experiments discussed, but as a general conclusion one can say that having more observations (the frequency of the observations sampled) better assimilation can be obtained. Another point is that for MP-NN the use of hyperbolic-tangent in the output layer improves the assimilation. Similar feature is found related to the RBF-NN, using Gaussian function in the output layer. Finally, an obvious point is that architectures with smaller number of neurons are preferred, from computational point of view.

For next, the experiments where ANNs were trained using 2000 examples, with the sampled observations at 0.01 s using 3, up to 40 neurons for both ANNs. The activation function implemented in the hidden layer is the hyperbolic-tangent, for all experiments, while in the output layer the activation function could be linear or hyperbolic-tangent, as shown in Table I. As before, the Gaussian function was used as activation function for the RBF-NN, while linear and Gaussian functions were used in the output layer – see Table I.

The experiments detect *overfiting* using 2000 patterns for training. The estimation presents a large **RMS** value, related to the experiments with 1000 patterns. Estimates with greater period of sampled observation result in a **RMS** greater than those obtained with sampled observation at 0.01 s.

Finally, results obtained using 4000 examples for training are analyzed. The same architecture for MP-NN and RBF-NN used before with 1000 and 2000 examples.

The *overfiting* problem was expected. However, estimations with 4000 examples produce better results than those obtained using 2000 patterns, but the assimilation is worse than 1000 patterns are used. Sampled observations at 0.1 s are indicating better results than those sampled at 0.01 s, for MP-NN.

Results for MP-NN with 2 hidden layers, having sampled observations at 0.01 s, present similar answer to those obtained with only one hidden layer. But, the computational cost for the NN with 2 hidden layer is greater than for a NN with one hidden layer. RBF-NN has one hidden layer, by definition, but some tests using 2 hidden layers were done with this topology, but bad results were obtained.

Conclusion

Artificial neural networks were applied in a assimilation process during the time integration of the Lorenz system in chaotic regime. Tests are done varying the size of the training set, and the time-period of the sampled observations inserted in the integration. The ANNs applied in the assimilation are the MP-NN and RBF-NN with different number of

neurons in the hidden layer(s). The performance of these NNs were also verified related to the use of linear and non-linear activation functions in the output layer.

It is a hard task to find out the optimum architecture for a given NN. However, this is not a constrain for its use, since the problem can be solved with a desired accuracy with simple architecture (few neurons and 1 or 2 layers), implying in a smaller computational cost related to the more complex (bigger) NNs. In few words, having good results, it is not necessary to find the optimum architecture.

The goal of the present study is not to do a formal analysis for each architecture, and the learning strategy employed, instead, the focus here is to show some general tendencies. From these general aspect, we can pointed out that for this application, the use of 1000 examples for training is clearly better than use 2000 or 4000 patterns. The result is better and the computational cost is smaller.

Concerning the time-period for sampled observations, it is important to note that greater time-period do not imply in a worse result, differently of the KF and other traditional schemes. In this work, the best results are obtained inserting observations at each 0.001 s, but greater or less quantity of observational data is characteristic for a given application. In meteorology, observational data are available at 12 and 24 h by the operational meteorological centers, such as NCEP (National Centers for Environmental Prediction) and ECMWF (European Certer for Medium-Range Weather Forecasts). Satellite data have also high interest for data assimilation.

Our experiments suggest that the use of hyperbolic-tangent in the output layer as the activation function of the MP-NN produces better results. The same strategy and conclusion can be applied to the RBF-NN, using the Gassian function in the output layer, since it present smaller sum of the square error.

A future work is to use recurrent neural network for data assimilation. This type of ANN is a system with memory, while the ANNs used in the present paper are memoryless system. Other features that motivate the study of the ANN for data assimilation is that ANNs are essentially parallel algorithms, and they can be implemented in hardware devices.

References

1. Zannetti, P. Air Pollution Modeling, Computational Mechanics Publications, UK, 1990.
2. Daley, R. Atmospheric Data Analysis, Cambridge University Press, Cambridge, 1991.
3. Bennet, A. F. Inverse Methods in Physical Oceanography, Cambridge University Press, 1992.
4. Nowosad, A. G. Data Assimilation Using an Adaptative Kalman Filter and Laplace Transform / A. G. Nowosad, A. Rios Neto, H.F. de Campos Velho // Hybrid Methods in Engineering. – 2000a. –Vol. 2.- P. 291-310.
5. Haykin, S. Neural Networks: A Comprehensive Foundation. Macmillan, New York.
6. Lorenz, E. Deterministic Nonperiodic Flow / E. Lorenz // Journal of Atmospheric Sciences. -1963. – Vol. 20 – P. 130-141.
7. Girosi, F. Networks and the best approximation property/ F. Girosi,T. Poggio // Biological Cibernetic. -1990. –Vol. 45 – P. 169-176. DOI: 10.1007/BF00195855
8. Jazwinski, A. Stochastic Processes and Filtering Theory, Academic Press, New York and London, 1970.

Fabrício Pereira Harter, professor, Faculty of Meteorology, Pelotas Federal University (Pelotas, RS, Brazil), fabricio.harter@ufpel.edu.br.

Haroldo Fraga de Campos Velho, research, Computing and Applied Mathematics, National Institute For Space Research (São José dos Campos, SP, Brazil), haroldo@lac.inpe.br.

Received April 20, 2014.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 96–108

УДК 551.509, 004.94 **DOI: 10.14529/cmse140407**

ИССЛЕДОВАНИЕ РАЗЛИЧНЫХ ВИДОВ ТОПОЛОГИИ НЕЙРОННЫХ СЕТЕЙ ДЛЯ АССИМИЛЯЦИИ ДАННЫХ

Ф.П. Хартер, Г.Ф. де Кампос Вельо

Методы нейронных сетей рассматриваются как альтернатива для существующих схем усвоения наблюдений в геофизические численные модели. Алгоритмы радиальных базисных функций и многослойного перцептрона выбраны для экспериментов по ассимиляции данных в простейшую двумерную гидродинамическую модель, т.н. систему динамического хаоса Лоренца. Обучение обоих типов алгоритмов производилось на выборке из 1000, 2000 и 4000 наблюдений поведения параметров системы с интервалами в 0.01, 0.06 и 0.1 сек, и затем в режиме распознавания произведена сравнительная оценка качества усвоения данных различными архитектурами нейронных сетей.

Ключевые слова: ассимиляция данных, нейронные сети.

Литература

1. Zannetti, P. Air Pollution Modeling, Computational Mechanics Publications, UK, 1990.
2. Daley, R. Atmospheric Data Analysis, Cambridge University Press, Cambridge, 1991.
3. Bennet, A. F. Inverse Methods in Physical Oceanography, Cambridge University Press, 1992.
4. Nowosad, A. G. Data Assimilation Using an Adaptative Kalman Filter and Laplace Transform / A. G. Nowosad, A. Rios Neto, H.F. de Campos Velho // Hybrid Methods in Engineering. – 2000a. –Vol. 2.- P. 291-310.
5. Haykin, S. Neural Networks: A Comprehensive Foundation. Macmillan, New York.
6. Lorenz, E. Deterministic Nonperiodic Flow / E. Lorenz // Journal of Atmospheric Sciences. -1963. – Vol. 20 – P. 130-141.
7. Girosi, F. Networks and the best approximation property/ F. Girosi,T. Poggio // Biological Cibernetic. -1990. –Vol. 45 – P. 169-176.
8. Jazwinski, A. Stochastic Processes and Filtering Theory, Academic Press, New York and London, 1970.

Фабрисио Перейра Хартер, преподаватель метеорологического факультета, Федеральный университет г. Пелотас, Бразилия.

Гарольдо Фрага де Кампос Вельо, исследователь вычислительной и прикладной математики, Национальный институт космических исследований, г. Сан-Жозе-дос-Кампос, Сан Пауло, Бразилия.

Поступила в редакцию 20 апреля 2014 г.

УДК 004.94 DOI: 10.14529/cmse140408

РАЗРАБОТКА СИСТЕМЫ РЕКОНСТРУКЦИИ НЕОДНОРОДНЫХ ТЕЛ ЭЛЕМЕНТАРНЫМИ ОБЪЕМАМИ НА ПРИМЕРЕ КЕРАМИКИ С ДЕФЕКТАМИ МИКРОСТРУКТУРЫ

М.О. Кибель, Н.Ю. Долганина

Работа посвящена разработке системы реконструкции неоднородных тел элементарными объемами на примере керамики с дефектами микроструктуры. Система была спроектирована и реализована. Проведено тестирование системы. Данная система позволяет создавать модели керамических структур с учетом распределения дефектов, которые предназначены для суперкомпьютерного моделирования ударного нагружения керамик в пакете программ LS-DYNA.

Ключевые слова: броня, керамика, дефекты, микроструктура, метод конечных элементов, элементарный объем, моделирование.

Введение

В настоящее время бронированная техника имеет, в основном, металлическую конструкцию (сталь, алюминиевые, титановые сплавы), что приводит к неоправданно высоким весовым показателям и необходимости иметь более мощные и тяжелые двигатели. Требования современного театра боевых действий заставляют разрабатывать легкие и скоростные транспортные средства с броней из керамики и полимерных волокнистых композитов [1, 2]. Однако количество варьируемых параметров при оптимизации защитной структуры исчисляется десятками [3–5], поэтому актуальной является разработка детализированных математических моделей деформирования и разрушения слоистых керамо-композитных структур для реализации конечно-элементных оптимизационных суперкомпьютерных вычислений.

Анализ современной литературы по механике разрушения неоднородных материалов показывает явную необходимость учета исходного состояния материала на микроуровне [1], т.е. учета дефектов структуры: распределения пор, включений и других несовершенств. Таким образом, многопараметричность моделей сплошной среды для керамики будет резко снижена за счет повышения «физичности и структурности» при отказе от гипотезы макросплошности. Перенос внимания на включение дефектов разного уровня в математические модели геометрии с упрощенным малопараметрическим описанием разрушения — мировой тренд численного анализа [6, 7].

Однако если для композитных материалов такой подход успешно применяется несколько лет [8–10], то в области моделирования керамических материалов есть всего несколько работ. Предлагаемые упрощенные аналитические модели [11] подходят только для первоначальных грубых оценок. Поэтому предлагается разработать систему реконструкции неоднородных тел элементарными объемами, где будет учтена микроструктура хрупкого керамического материала, полученные модели керамики будут использоваться при конечно-элементном моделировании процессов высокоскоростного удара на суперкомпьютере в пакете программ LS-DYNA.

Статья организована следующим образом. В разделе 1 приведена постановка задачи. В разделе 2 описывается реализация системы. В разделе 3 обсуждаются результаты исследований. В заключении суммируются основные результаты, полученные в данной работе.

1. Постановка задачи

1.1. Функциональные требования к системе

Система должна предоставлять пользователю графический интерфейс со следующими возможностями:

- ввод имени входного текстового файла с граничными условиями (координаты точек);

- выбор количества измерений модели (2D или 3D);

- ввод размера конечного элемента (в двухмерном случае в качестве формы конечного элемента берется квадрат, в трехмерном — куб);

- ввод процентного соотношения нескольких материалов (например, 20 %; 15 %; 5 %; 60 %). При генерации сетки конечных элементов каждому конечному элементу в модели случайным образом присваивается один из нескольких заданных материалов, в результате на выходе получается модель с распределенными случайным образом свойствами материалов по объему.

Система должна обеспечивать создание k-файла (входного файла для расчета в пакете программ LS-DYNA) с узлами и конечными элементами тела в 2D или 3D постановке, реконструированного в соответствии с выбранными параметрами.

1.2. Модульная структура системы

Система имеет модульную структуру:

- *Модуль распределения свойств* предназначен для распределения свойств материалов конечных элементов случайным образом по объему тела. Процентное соотношение материалов задано пользователем.

- *Модуль анализа координат границ* предназначен для обработки границ реконструируемого тела. Он выполняет следующие функции: считывание координат точек границы из входного файла; обработка границ; представление границ в системе.

- *Модуль построения сетки* предназначен для преобразования тела в набор конечных элементов. Он выполняет следующие функции: разбиение фигуры; анализ положения конечных элементов относительно границ фигуры.

- *Модуль вывода* предназначен для записи вычисленных координат узлов и конечных элементов в выходной k-файл. Данные в *модуль вывода* поступают из *модуля построения сетки* и *модуля распределения свойств*.

1.3. Взаимодействие между модулями

На рис. 1 представлена схема взаимодействия между модулями системы. Стрелками обозначены потоки данных от одного модуля к другому. Основная программа предоставляет пользователю интерфейс для ввода параметров вычисления. Эти параметры обрабатываются и передаются в *модуль распределения свойств* и *модуль анализа координат границ* соответственно.

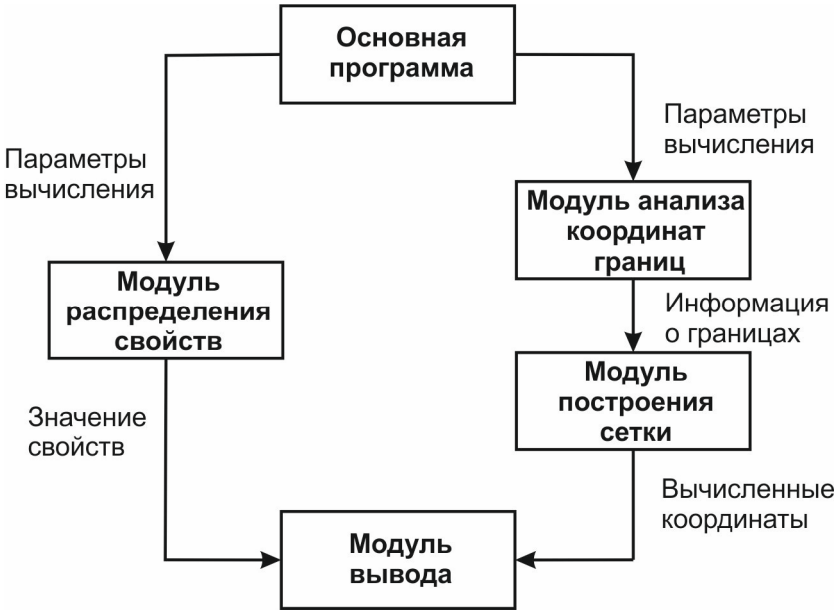

Рис. 1. Схема взаимодействия модулей системы

2. Реализация системы

Пользовательский интерфейс (рис. 2) представляет собой форму ввода данных, необходимых для реконструкции тела. Для работы с приложением пользователю необходимо ввести имя входного файла, выбрать необходимое количество измерений, внести размер конечного элемента и процентное соотношение материалов. После этого нажать кнопку «Вычислить» и система построит модель на основе введенных параметров.

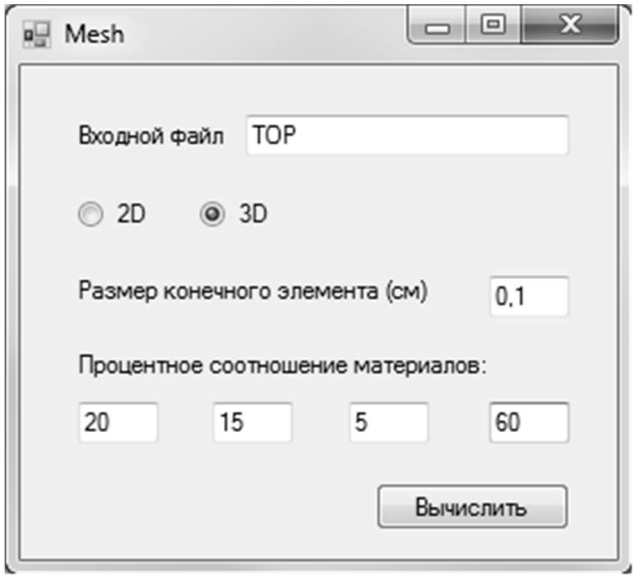

Рис. 2. Пользовательский интерфейс

Алгоритм работы системы включает в себя следующие шаги:

1. Считывание координат границ из входного текстового файла. После считывания координаты записываются в vector (шаблон из стандартной библиотеки C++, реализующий динамический массив произвольного доступа), где с ними происходит дальнейшая обработка.

2. Обработка границ и представление их в системе в виде конечных элементов.

3. Вписывание тела в прямоугольник в двухмерном случае и в параллелепипед в трехмерном, на основе вычисленных крайних точек тела: минимумов и максимумов по каждой из осей.

4. Заполнение полученного параллелепипеда конечными элементами.

5. Определение принадлежности каждого конечного элемента реконструируемому телу. Для этого используется метод трассировки луча с учетом числа пересечений [12]. Из текущего конечного элемента выпускается луч по трем осям, по каждому подсчитывается число пересечений с границами, исходя из полученных данных, определяется положение конечного элемента относительно границ.

6. Удаление конечных элементов, не вошедших в тело.

7. Распределение свойств материалов конечных элементов.

8. Запись вычисленных узлов и конечных элементов в k-файл.

3. Результаты исследований

Для тестирования системы использовались различные выпуклые и невыпуклые двухмерные и трехмерные тела. На рис. 3 представлен один из тестов: разбиение тора на конечные элементы с процентным соотношением материалов 20 %, 15 %, 5 %, 60 %.

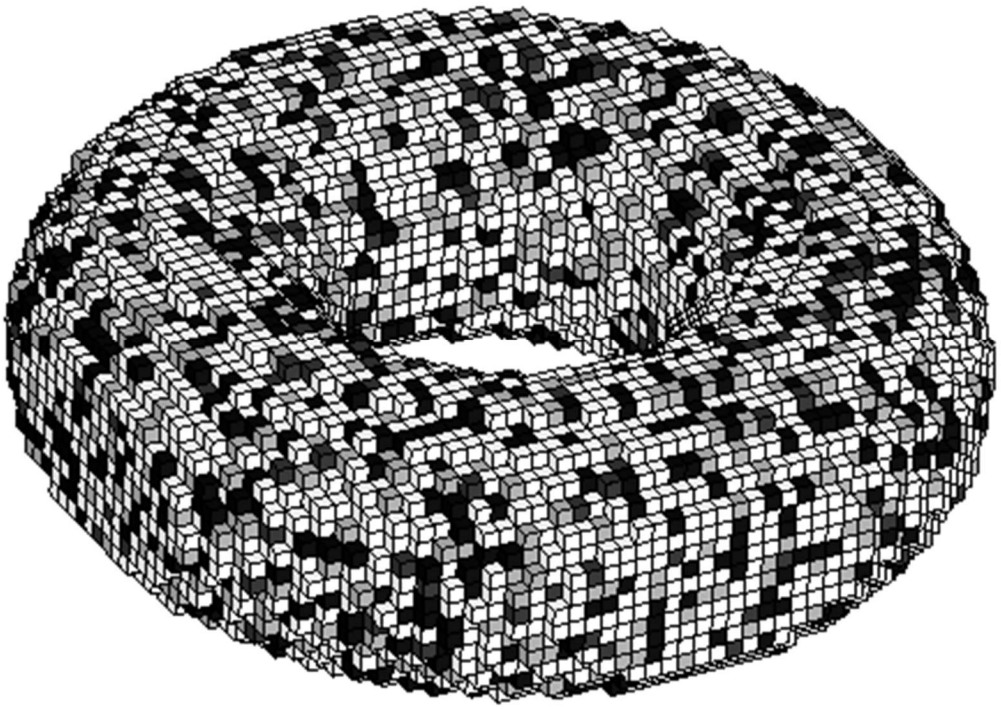

Рис. 3. Разбиение тора на конечные элементы с процентным соотношением материалов: 20 % (черный), 15 % (светло серый), 5 % (темно серый), 60 % (белый)

Заключение

Разработана система реконструкции неоднородных тел элементарными объемами на примере керамики с дефектами микроструктуры. В ходе работы были решены следующие задачи: спроектирована и реализована система реконструкции неоднородных тел элементарными объемами, проведено тестирование системы на различных выпуклых и

невыпуклых двухмерных и трехмерных телах. Система позволяет создавать модели керамических структур с учетом дефектов. Полученные модели будут использоваться для суперкомпьютерного моделирования ударного нагружения керамик в пакете программ LS-DYNA.

Исследование выполнено в Южно-Уральском государственном университете (национальном исследовательском университете) за счет гранта Российского научного фонда (проект № 14-19-00327).

Литература

1. Hazell, P.J. Ceramic Armour: Design and Defeat Mechanisms / P.J. Hazell — Canberra: Argos Press, 2006. — 168 p.
2. Bhatnagar, A. Lightweight Ballistic Composites. Military and Law-Enforcement Applications / A. Bhatnagar — Cambridge: Woodhead publishing limited, 2006. — 429 p.
3. Krishnan, K. Numerical Simulation of Ceramic Composite Armor Subjected to Ballistic Impact / K. Krishnan, S. Sockalingam, S. Bansal, S.D. Rajan // Composites. — 2010. — Part B 41. — P. 583–593.
4. Feli, S. Finite Element Simulation of Ceramic/Composite Armor under Ballistic Impact / S. Feli, M.R. Asgari // Composites. — 2011. — Part B: Engineering, Vol. 42, Issue 4. — P. 771–780.
5. Bürgera, D. Ballistic Impact Simulation of an Armour-Piercing Projectile on Hybrid Ceramic/Fiber Reinforced Composite Armours / D. Bürgera, A.R. Fariab, S.F.M. Almeidab, F.C.L. Meloa, M.V. Donadonb // International Journal of Impact Engineering. — 2012. — Vol. 43. — P. 63–77.
6. Swab, J.J. Advances in Ceramic Armor VII. A Collection of Papers Presented at the 35nd International Conference on Advanced Ceramics and Composites / J.J. Swab — Hoboken: John Wiley & Sons, 2011. — 272 p.
7. Sapozhnikov, S.B. Ballistic Damage, Residual Strength and Repair of GFRP Plates / S.B. Sapozhnikov, M.V. Zhikharev // 1st International Conference for Advanced Marine Engineering (ICACME 2013). — Sept. 2013. — P. 28.
8. Долганина, Н.Ю. Проектирование новых конструкций тканевых бронепанелей с использованием суперкомпьютерных вычислений / Н.Ю. Долганина, С.Б. Сапожников // Вестник Южно-Уральского государственного университета. Серия «Математическое моделирование и программирование». — 2011. — № 37(254). — С. 71–81.
9. Долганина, Н.Ю. Исследование ударного взаимодействия индентора с тканевыми бронепластинами, расположенными на пластилиновом основании / Н.Ю. Долганина // Вестник Южно-Уральского государственного университета. Серия «Вычислительная математика и информатика». — 2012. — № 47(306). — С. 37–45.
10. Долганина, Н.Ю. Исследование влияния типа переплетения нитей на прочность тканевых преград при локальном ударе / Долганина Н.Ю., Сапожников С.Б. // Вестник Южно-Уральского государственного университета. Серия «Машиностроение». — 2013. — № 2. — С. 95–104.
11. Сапожников, С.Б. Особенности разрушения пластины из хрупкого материала при взаимодействии с ударником / С.Б. Сапожников, О.А. Кудрявцев // Вестник Юж-

но-Уральского государственного университета. Серия «Математика. Механика. Физика». — 2012. — № 34(293). — С. 177–181.

12. Препарата, Ф. Вычислительная геометрия. Введение / Ф. Препарата, М. Шеймос — М.: Мир, 1989. — 478 с.

Кибель Мария Олеговна, магистрант, Южно-Уральский государственный университет (Челябинск, Российская Федерация), mary.kibel@gmail.com.

Долганина Наталья Юрьевна, к.т.н., доцент кафедры системного программирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), dolganinani@susu.ac.ru.

Поступила в редакцию 5 августа 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 109–115

DOI: 10.14529/cmse140408

DEVELOPMENT SYSTEM FOR RECONSTRUCTION INHOMOGENEOUS BODIES BY ELEMENTARY VOLUMES ON THE EXAMPLE OF THE CERAMICS WITH DEFECTS OF MICROSTRUCTURE

M.O. Kibel, South Ural State University (Chelyabinsk, Russian Federation),
N.Yu. Dolganina, South Ural State University (Chelyabinsk, Russian Federation)

Work is devoted to the development a system for reconstruction inhomogeneous bodies by elementary volumes on the example of the ceramics with defects of microstructure. The system was designed and implemented. System testing was performed. This system allows to create models of ceramic structures with the defect distribution. The resulting models are designed for the supercomputer simulation shock loading of ceramics in the framework of the LS-DYNA software package.

Keywords: armor, ceramics, defects, microstructure, finite element method, the elementary volume, modeling.

References

1. Hazell P.J. Ceramic Armour: Design and Defeat Mechanisms. Canberra: Argos Press, 2006. 168 p.
2. Bhatnagar A. Lightweight Ballistic Composites. Military and Law-Enforcement Applications. Cambridge: Woodhead publishing limited, 2006. 429 p.
3. Krishnan K., Sockalingam S., Bansal S., Rajan S.D. Numerical Simulation of Ceramic Composite Armor Subjected to Ballistic Impact // Composites. 2010. Part B 41. P. 583–593. DOI: 10.1016/j.compositesb.2010.10.001.

4. Feli, S., Asgari M.R. Finite Element Simulation of Ceramic/Composite Armor under Ballistic Impact // Composites. 2011. Part B: Engineering, Vol. 42, Issue 4. P. 771–780. DOI: 10.1016/j.compositesb.2011.01.024.
5. Bürgera D., Fariab A.R., Almeidab S.F.M., Meloa F.C.L., Donadonb M.V. Ballistic Impact Simulation of an Armour-Piercing Projectile on Hybrid Ceramic/Fiber Reinforced Composite Armours // International Journal of Impact Engineering. 2012. Vol. 43. P. 63–77. DOI: 10.1016/j.ijimpeng.2011.12.001.
6. Swab J.J. Advances in Ceramic Armor VII. A Collection of Papers Presented at the 35th International Conference on Advanced Ceramics and Composites. Hoboken: John Wiley & Sons, 2011. 272 p.
7. Sapozhnikov S.B., Zhikharev M.V. Ballistic Damage, Residual Strength and Repair of GFRP Plates // 1st International Conference for Advanced Marine Engineering (ICACME 2013). Sept. 2013. P. 28.
8. Dolganina N.Yu., Sapozhnikov S.B. Proyektirovaniye novykh konstruktsiy tkanevykh bronepaneley s ispolzovaniyem superkompyuternykh vychisleniy [Design of New Constructions of Textile Armor Panel Using Supercomputing] // Vestnik Yuzhno-Uralskogo gosudarstvennogo universiteta. Seriya «Matematicheskoye modelirovaniye i programmirovaniye» [Bulletin of the South Ural State University. Series «Mathematical Modelling, Programming & Computer Software»]. 2011. No. 37(254). P. 71–81.
9. Dolganina N.Yu. Issledovaniye udarnogo vzaimodeystviya indentora s tkanevymi broneplastinami, raspolozhennymi na plastilinovom osnovanii [Investigation of Impact Interaction of the Indenter with Fabric Armor Plate Which Located on the Clay Basis] // Vestnik Yuzhno-Uralskogo gosudarstvennogo universiteta. Seriya «Vychislitelnaya matematika i informatika» [Bulletin of the South Ural State University. Series «Computational Mathematics and Software Engineering»]. 2012. No. 47(306). P. 37–45.
10. Dolganina N.Yu., Sapozhnikov S.B. Issledovaniye vliyaniya tipa perepleteniya nitey na prochnost tkanevykh pregrad pri lokalnom udare [Study of the Influence of Type Weave for Strength of the Textile Armor Panel at the Local Impact] // Vestnik Yuzhno-Uralskogo gosudarstvennogo universiteta. Seriya «Mashinostroyeniye» [Bulletin of the South Ural State University. Series «Mechanical Engineering Industry»]. 2013. No. 2. P. 95–104.
11. Sapozhnikov S.B., Kudryavtsev O.A. Osobennosti razrusheniya plastiny iz khrupkogo materiala pri vzaimodeystvii s udarnikom [Aspects of Brittle Plate Fracture Due to Interaction with Indenter] // Vestnik Yuzhno-Uralskogo gosudarstvennogo universiteta. Seriya «Matematika. Mekhanika. Fizika» [Bulletin of the South Ural State University. Series «Mathematics. Mechanics. Physics»]. 2012. No. 34(293). P. 177–181.
12. Preparata F., Sheymos M. Vychislitelnaya geometriya. Vvedeniye [Computational Geometry. Introduction]. M.: Mir, 1989. 478 p.

Received August 5, 2014 г.

УДК 519.853.4 DOI: 10.14529/cmse140409

ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНЫХ ХАРАКТЕРИСТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ МНОГОМЕРНЫХ ЗАДАЧ ГЛОБАЛЬНОЙ ОПТИМИЗАЦИИ[1]

К.А. Баркалов

В статье изложены результаты исследования многоуровневой схемы редукции размерности в задачах глобальной оптимизации. Предложенная схема позволяет свести решение многомерной задачи оптимизации к серии подзадач меньшей размерности, решение которых может быть выполнено параллельно. При этом для редукции размерности комбинируется использование кривых Пеано и схема вложенной (рекурсивной) оптимизации. Для решения редуцированных подзадач используется параллельный алгоритм глобального поиска, принадлежащий классу характеристических алгоритмов. Проведены вычислительные эксперименты на серии тестовых задач разной размерности. Результаты экспериментов показывают, что предложенная схема позволяет эффективно распараллелить процесс поиска и добиться значительного ускорения.

Ключевые слова: глобальная оптимизация, многоэкстремальные функции, редукция размерности, характеристические алгоритмы, параллельные алгоритмы.

Введение

Рассматриваются задачи многоэкстремальной (глобальной) оптимизации вида

$$\varphi(y^*) = \min\{\varphi(y): y \in D\}, \tag{1}$$

$$D = \{y \in R^N: a_i \le y_i \le b_i, 1 \le i \le N\}. \tag{2}$$

Важный в прикладном отношении подкласс задач (1) характеризуется тем, что минимизируемая функция задана некоторым (программно реализуемым) алгоритмом вычисления значений $\varphi(y)$ в точках области поиска D. В этом случае аналитические способы решения не применимы, а численное решение задачи сводится к построению оценки точного решения на основе некоторого числа k значений функций задачи, вычисленных в точках области D. Процесс вычисления значения функции в точке будем назвать испытанием.

В задачах многоэкстремальной оптимизации возможность достоверной оценки глобального оптимума принципиально основана на наличии априорной информации о функции, позволяющей связать возможные значения минимизируемой функции с известными значениями в точках осуществленных поисковых итераций. Весьма часто такая априорная информация о задаче (1) представляется в виде предположения, что целевая функция φ удовлетворяет условию Липшица с константой L. Методам решения задач данного класса посвящена обширная литература (см. обзоры в [1–3]). Обзор прикладных проблем глобальной оптимизации приведен в [4].

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

Использование современных многопроцессорных вычислительных систем расширяет границы применения методов глобальной оптимизации и в то же время ставит задачу эффективного распараллеливания процесса поиска. Отметим, что распараллеливание процесса поиска можно проводить в нескольких направлениях. Во-первых, можно распараллеливать вычисление целевой функции, во-вторых, реализацию вычислительных правил алгоритма, обеспечивающих выбор точки проведения очередного испытания, и, в-третьих, можно изменить схему алгоритма с целью параллельного выполнения нескольких испытаний. Однако распараллеливание вычисления целевой функции, описывающей оптимизируемый объект, специфично для конкретной решаемой задачи. Распараллеливание правил алгоритма существенно зависит от конкретного класса алгоритмов и, кроме того, часто эти правила достаточно просты и распараллеливать их нецелесообразно. Поэтому перспективным является третий подход, который характеризуется эффективностью (распараллеливается именно та часть вычислительного процесса, в котором выполняется основной объем вычислений) и общностью (применим для широкого класса характеристических алгоритмов многоэкстремальной оптимизации).

Так, в работе [5] введен и исследован класс параллельных характеристических алгоритмов; установлена преемственность свойств сходимости параллельных характеристических алгоритмов по отношению к последовательным прототипам, оценена избыточность параллельных характеристических алгоритмов. Следуя этим оценкам, параллельные характеристические алгоритмы являются эффективными при небольшом числе (не более четырех) используемых процессоров. Использование же нескольких сотен параллельных вычислительных элементов не даст ускорения процесса поиска в несколько сотен раз. Причина состоит в том, что в этом случае параллельный алгоритм будет порождать значительное число избыточных точек испытаний (по сравнению с последовательным прототипом). В данной работе исследована новая схема использования параллельных характеристических алгоритмов, позволяющая обойти данную проблему и обеспечить значительное ускорение при решении многомерных многоэкстремальных задач.

1. Редукция размерности

Для снижения сложности алгоритмов глобальной оптимизации, формирующих эффективные покрытия области поиска, в многоэкстремальной оптимизации широко используются различные схемы редукции размерности, которые позволяют свести решение многомерных оптимизационных задач к семейству задач одномерной оптимизации. Редукция размерности позволяет существенно снизить сложность разрабатываемых алгоритмов глобального поиска. Кроме того, данный подход позволяет задействовать весь имеющийся аппарат одномерной многоэкстремальной оптимизации для построения эффективных многомерных методов глобального поиска.

Известно два общих метода редукции размерности. Один из них состоит в применении разверток единичного отрезка вещественной оси на гиперкуб. Роль таких разверток играют непрерывные однозначные отображения типа кривой Пеано, называемые также кривыми, заполняющими пространство. Использование развертки Пеано $y(x)$, однозначно отображающей единичный отрезок вещественной оси на единичный гиперкуб, позволяет свести многомерную задачу минимизации в области D к одномерной задаче минимизации на отрезке [0,1]

$$\varphi(y^*) = \varphi(y(x^*)) = \min\{\varphi(y(x)) : x \in [0,1]\}. \quad (3)$$

Вопросы численного построения отображений типа кривой Пеано и соответствующая теория подробно рассмотрены в [1, 3].

Второй способ редукции размерности – схема вложенной оптимизации, или же многошаговая схема – основана на соотношении

$$\min_{y \in D} \varphi(y) = \min_{a_1 \le y_1 \le b_1} \min_{a_2 \le y_2 \le b_2} \dots \min_{a_N \le y_N \le b_N} \varphi(y), \quad (4)$$

которое сводит решение исходной многомерной задачи к решению семейства рекурсивно связанных одномерных подзадач. При этом каждое вычисление целевой функции одномерной подзадачи есть решение новой одномерной подзадачи следующего уровня рекурсии, за исключением последнего уровня, где вычисляется значения исходной многомерной целевой функции. Различные модификации многошаговой схемы и соответствующая теория сходимости представлены в [3].

Для решения возникающих одномерных задач как в первом, так и во втором подходах могут быть использованы параллельные характеристические алгоритмы глобального поиска [5]. Способ распараллеливания в алгоритмах данного класса основан на следующем соображении: в характеристических алгоритме глобального поиска каждому поисковому интервалу сопоставляется некоторое число (характеристика интервала), которое может рассматриваться как мера важности данного интервала для последующего поиска решения в нем. Следуя данному пониманию, для организации параллельных вычислений после выбора точки вычисления значения функции для первого процессора в точном соответствии с последовательным алгоритмом, для второго процессора точку очередной итерации целесообразно выбирать из следующего по важности интервала (т.е. из интервала со следующей по порядку максимальной характеристикой) и т.д.

2. Многоуровневая схема редукции размерности

Для изложенной выше многошаговой схемы предложено обобщение (блочная многошаговая схема), которое комбинирует использование разверток типа кривой Пеано и многошаговой схемы с целью эффективного распараллеливания вычислений.

Рассмотрим вектор y как вектор «агрегированных» макро-переменных

$$y = (y_1, y_2, \dots, y_N) = (u_1, u_2, \dots, u_M), \quad (5)$$

где i-я макропеременная u_i представляет собой вектор размерности N_i из последовательно взятых компонент вектора y. С использованием макро-переменных основное соотношение многошаговой схемы (4) может быть переписано в виде

$$\min_{y \in D} \varphi(y) = \min_{u_1 \in D_1} \min_{u_2 \in D_2} \dots \min_{u_M \in D_M} \varphi(y), \quad (6)$$

где подобласти $D_i, 1 \le i \le M$, являются проекциями исходной области поиска D на подпространства, соответствующие макро-переменным $u_i, 1 \le i \le M$.

Формулы, определяющие способ решения задачи (1) на основе соотношения (6) полностью совпадают с многошаговой схемой (4). Требуется лишь заменить исходные переменные $y_i, 1 \le i \le N$, на макро-переменные $u_i, 1 \le i \le M$. При этом принципиальным отличием от исходной схемы является тот факт, что в блочной многошаговой схеме вложенные подзадачи являются многомерными, и для их решения может быть применен способ редукции размерности на основе кривых Пеано. Число векторов и количество компонент в каждом векторе являются параметрами блочной многошаговой схемы и

могут быть использованы для формирования подзадач с нужными свойствами, и здесь возможны два диаметрально противоположных случая.

Если $M = N$, то блочная многошаговая схема идентична исходной многошаговой схеме; каждая из вложенных подзадач является одномерной.

Если $M = 1$, то решение задачи эквивалентно ее решению с использованием единственной кривой Пеано, отображающей [0,1] в D; вложенные подзадачи отсутствуют.

Нас же будет интересовать промежуточный вариант $1 < M \ll N$, при котором исходная задача большой размерности разбивается на 2-3 вложенные подзадачи меньшей размерности. Тогда, применяя в многошаговой схеме для решения вложенных подзадач параллельные характеристические методы глобальной оптимизации, мы получим параллельный вариант блочной многошаговой схемы с широкой степенью вариативности. Например, можно варьировать количество процессоров на различных уровнях оптимизации (т.е. при решении подзадач по различным макро-переменным), применять различные параллельные методы поиска на разных уровнях и т.п.

Для описания процесса параллелизма многошаговой схемы (следуя [6]) введем вектор степеней распараллеливания $\pi = (\pi_1, \pi_2, \ldots, \pi_M)$, где $\pi_i, 1 \le i \le M$, обозначает число параллельно решаемых подзадач (i+1)-го уровня вложенности, возникающих в результате выполнения параллельных итераций на i-м уровне.

В общем случае величины $\pi_i, 1 \le i \le M$, могут зависеть от различных параметров и меняться в процессе оптимизации, но мы ограничимся случаем, когда все компоненты вектора π постоянны. Применение параллельных характеристических алгоритмов в соответствии с блочной многошаговой схемой позволяет использовать для решения исходной задачи

$$\Pi = \prod_{i=1}^{M} \pi_i \tag{7}$$

параллельно работающих процессоров. Например, при использовании $\pi_i = 8, 1 \le i \le M$, общее число подзадач, для которых возможно параллельное решение, будет увеличиваться на каждом уровне в 8 раз, что при использовании всего трех макропеременных составит 512. Таким образом, предложенный способ распараллеливания позволяет эффективно задействовать сотни и тысячи узлов современных вычислительных систем.

3. Результаты экспериментов

В данном разделе представлены результаты вычислительных экспериментов. Изложенный выше параллельный алгоритм глобальной оптимизации с использованием многоуровневой схемы редукции размерности был реализован в виде MPI-программы на языке C++. Все эксперименты были проведены на вычислительном кластере ННГУ. Формулировки и названия тестовых задач взяты из [7, 8].

Вначале проведем исследование эффективности последовательного алгоритма на четырехмерной задаче Shekel 10. В табл. 1 приведены результаты решения данной задачи: число выполненных испытаний K, время решения T в секундах, найденное значение оптимума φ^*. Во всех экспериментах использовались параметры метода $r = 2.8$, $\varepsilon = 0.002$ и параметр построения кривой Пеано $m = 10$; варьировалось лишь число M и размерность N_i макро-переменных u_i.

Таблица 1

Результаты решения задачи Shekel 10

M	N_i	K	T	φ^*
1	4	64 604	0.65	–10.53
2	2,2	140 925	0.89	–10.53
2	3,1	639 780	3.08	–10.52
2	1,3	399 960	3.44	–10.52
3	1,1,2	1 405 598	8.87	–10.52
3	1,2,1	5 863 336	27.7	–10.51
3	2,1,1	3 045 636	14.56	–10.52
4	1,1,1,1	18 748 757	88.8	–10.52

Данные результаты подтверждают, что наименьшее число поисковых испытаний выполняет метод при $M = 1$. Однако в данном случае метод будет плохо распараллеливаться (см. обсуждение выше), поэтому эффективность распараллеливания будем исследовать при $M = 2$ и $N_1 = N_2 = 2$. Так как значение функции вычисляется быстро, то вектор распараллеливания π будем выбирать в виде $(p–1,1)$, где p — число процессов в MPI-программе (последовательный запуск программы обозначен $(0,0)$). Ниже приведены результаты решения нескольких задачи из [8]: в табл. 2 — время решения в секундах, в табл. 3 — число вычислений значений оптимизируемой функции. Также приведены используемые параметры метода r и ε и найденное значение оптимума φ^*.

Таблица 2

Результаты решения четырехмерных задач

Problem	r	ε	φ^*	π=(0,0)	π=(2,1)	π=(4,1)	π=(8,1)
Levy 1	2	0.002	0.0003	13.33	6.3	3.62	3.01
Neumaier 2	2	0.01	0.005	31.3	12.1	9.1	7.38
Neumaier 3	3.2	0.002	–15.998	27.5	21.2	15.54	11.63
Rastrigin	2.5	0.002	0.003	2.2	2.96	1.5	0.7
Rosenbrock	2.1	0.01	0.1	14.22	23.31	9.13	6.7
Shekel 5	2.5	0.002	–10.152	0.92	0.35	0.19	0.18
Shekel 7	2.5	0.002	–10.402	0.72	0.44	0.25	0.16
Shekel 10	2.5	0.002	–10.534	0.65	0.54	0.15	0.23

Таблица 3

Результаты решения четырехмерных задач

Problem	r	ε	φ^*	π=(0,0)	π=(2,1)	π=(4,1)	π=(8,1)
Levy 1	2	0.002	0.0003	1378380	2145661	2021384	2349743
Neumaier 2	2	0.01	0.005	3059379	4870091	6976209	6252963
Neumaier 3	3.2	0.002	–15.998	2763317	8636710	11466074	10176447
Rastrigin	2.5	0.002	0.003	235766	1275904	1286144	999423
Rosenbrock	2.1	0.01	0.1	1491435	9309296	7206413	9499122
Shekel 5	2.5	0.002	–10.152	95302	122831	105876	127946
Shekel 7	2.5	0.002	–10.402	75820	146677	135331	131481
Shekel 10	2.5	0.002	–10.534	64604	181132	71182	154927

Результаты экспериментов показывают, что последовательный алгоритм с использованием одной кривой Пеано выполняет значительно меньше поисковых испытаний, чем алгоритм с редукцией размерности по многоуровневой схеме. В силу этого в некоторых случаях получается, что параллельный алгоритм с использованием вектора распараллеливания (2,1) и, соответственно, 3 MPI-процессов работает примерно так же (или даже медленнее) чем последовательный. Однако уже использование вектора распараллеливания (4,1) дает значительное ускорение. При этом можно сказать, что по времени работы и по числу выполненных поисковых испытаний при решении данных задач предложенный алгоритм не уступает другим известным методам [8].

Следующую серию экспериментов проведем на задаче Rastrigin из [8] при $N = 10$, $D = \{-2.2 \le y_i \le 1.8, 1 \le i \le N\}$. Глобально-оптимальное решение — $y^* = (0,\ldots,0)$, $\varphi(y^*) = 0$. В табл. 4 приведены результаты решения данной задачи: время решения T в секундах и найденное значение оптимума φ^*. Во всех экспериментах использовались параметры метода $r = 2.3$, $\varepsilon = 0.01$, параметр построения кривой Пеано $m = 10$. Число макро-переменных и их размерность были выбраны $M = 2$, $N_1 = 6, N_2 = 4$, т.е. решение исходной 10-мерной задачи сводится к решению 6-мерной, причем вычисление одного значения в ней подразумевает решение вложенной 4-мерной подзадачи. Варьировался вектор распараллеливания π, в соответствии с которым изменялось число процессов в MPI-программе от 9 до 101 соответственно.

Таблица 4

Результаты решения десятимерной задачи

π	T	φ^*
(8,1)	15 938	0.26
(20,1)	5 448	0.18
(40,1)	2 911	0.19
(70,1)	1 453	0.27

Во всех экспериментах число поисковых испытаний изменялось незначительно и составило порядка 10^9. Однако при этом произошло значительное сокращение времени решения задачи за счет увеличения числа используемых процессоров.

Заключение

В данной работе исследовалась многоуровневая схема редукции размерности в задачах глобальной оптимизации, комбинирующая использование кривых Пеано и многошаговую схему. Для решения редуцированных подзадач используется алгоритм глобального поиска, принадлежащий классу характеристических алгоритмов. Для предложенной многоуровневой схемы рассмотрены вопросы ее эффективного использования для задач с разным временем вычисления оптимизируемой функции.

Проведены вычислительные эксперименты на серии тестовых задач разной размерности с целью сравнения ускорения. Результаты экспериментов показывают, что предложенная схема позволяет эффективно распараллелить процесс поиска и добиться значительного ускорения. При этом разработанный алгоритм как минимум не уступает, а в некоторых случаях и превосходит известные методы аналогичного назначения.

Работа частично поддержана грантом МОН РФ (соглашение от 27 августа 2013 г. № 02.B.49.21.0003 между МОН РФ и ННГУ).

Литература

1. Strongin, R.G. Global Optimization with Non-convex Constraints. Sequential and Parallel Algorithms / R.G. Strongin, Ya.D. Sergeyev — Kluwer Academic Publishers, 2000. — 704 p.
2. Floudas, C.A. A Review of Recent Advances in Global Optimization / C.A. Floudas, C.E. Gounaris // Journal of Global Optimization. — 2009. — Vol. 45, No. 1. — P. 3–38.
3. Стронгин, Р.Г. Параллельные вычисления в задачах глобальной оптимизации / Р.Г. Стронгин, В.П. Гергель, В.А. Гришагин, К.А. Баркалов — М.: Издательство Московского университета, 2013. — 280 с.
4. Pinter, J.D. Global Optimization: Scientific and Engineering Case Studies / J.D. Pinter — Springer, 2006. — 546 p.
5. Grishagin, V.A. Parallel Characteristical Algorithms for Solving Problems of Global Optimization / V.A. Grishagin, Ya.D. Sergeyev, R.G. Strongin // Journal of Global Optimization. — 1997. — Vol. 10, No. 2. — P. 185–206.
6. Sergeyev, Ya.D. Parallel Asynchronous Global Search ant the Nested Optimization Scheme / Ya.D. Sergeyev, V.A. Grishagin // Journal of Computational Analysis and Applications. — 2001. — Vol. 3, No. 2. — P. 123–245.
7. Ali, M. A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems / M. Ali, Ch. Khompatraporn, Z.B. Zabinsky // Journal of Global Optimization. — 2005. — Vol. 31, No. 4. — P. 635–672.
8. Paulavicius, R. Parallel Branch and Bound for Global Optimization with Combination of Lipschitz Bounds / R. Paulavicius, J. Zilinskas, A. Grothey // Optimization Methods & Software. — 2011. — Vol. 26, No. 3. — P. 487–498.

Баркалов Константин Александрович, к.ф.-м.н., доцент кафедры математического обеспечения ЭВМ факультета ВМК, Нижегородский государственный университет им. Н.И. Лобачевского (Нижний Новгород, Российская Федерация), barkalov@fup.unn.ru.

Поступила в редакцию 11 августа 2014 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2014, vol. 3, no. 4, pp. 116–123

DOI: 10.14529/cmse140409

USE OF THE PARALLEL CHARACTERISTICAL ALGORITHMS FOR SOLVING MULTIVARIATE PROBLEMS OF GLOBAL OPTIMIZATION

K.A. Barkalov, N.I. Lobachevsky State University of Nizhni Novgorod (Nizhni Novgorod, Russian Federation)

In this paper the problems of multidimensional multiextremal optimization and multilevel scheme of dimension reduction are considered. The proposed scheme allows to reduce solution of multidimensional problems to solution of a number of subproblems with less dimension, which can be solved in parallel. The multilevel scheme combines the ideas of Peano-type space filling curves and nested optimization. To solve the reduces subproblems the parallel characteristical algorithm is used. Results of numerical experiments confirm convergence and speedup of the parallel algorithm.

Keywords: global optimization, multiextremal functions, dimension reduction, characteristical algorithms, parallel algorithms.

References

1. Strongin R.G., Sergeyev Ya.D. Global Optimization with Non-convex Constraints. Sequential and Parallel Algorithms. Kluwer Academic Publishers, 2000. 704 p.
2. Floudas C.A., Gounaris C.E. A Review of Recent Advances in Global Optimization // Journal of Global Optimization. 2009. Vol. 45, No. 1. P. 3–38. DOI: 10.1007/s10898-008-9332-8.
3. Strongin R.G., Gergel V.P., Grishagin V.A., Barkalov K.A. Parallelnie vichisleniya v zadachah globalnoi optimizatsii [Parallel Computing for Global Optimization Problems]/ Moscow: Moscow University Press, 2013. 280 p.
4. Pinter J.D. Global Optimization: Scientific and Engineering Case Studies. Springer, 2006. 546 p.
5. Grishagin V.A., Sergeyev Ya.D., Strongin R.G. Parallel Characteristical Algorithms for Solving Problems of Global Optimization // Journal of Global Optimization. 1997. Vol. 10, No. 2. P. 185–206.
6. Sergeyev Ya.D., Grishagin V.A. Parallel Asynchronous Global Search ant the Nested Optimization Scheme // Journal of Computational Analysis and Applications. 2001. Vol. 3, No. 2. P. 123–245. DOI: 10.1023/A:1010185125012.
7. Ali M., Khompatraporn Z.B., Zabinsky Z.B. A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems // Journal of Global Optimization. 2005. Vol. 31, No. 4. P. 635–672. DOI: 10.1007/s10898-004-9972-2.
8. Paulavicius R., Zilinskas J., Grothey A. Parallel Branch and Bound for Global Optimization with Combination of Lipschitz Bounds // Optimization Methods & Software. 2011. Vol. 26, No. 3. P. 487–498. DOI: 10.1007/978-0-387-09707-7_8.

Received August 11, 2014.

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.**