ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2015
Т. 4, № 4

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

Решением ВАК включен в Перечень научных изданий, в которых должны быть опубликованы результаты диссертаций на соискание ученых степеней кандидата и доктора наук

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

Тематика журнала:

- 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
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY

2015
Vol. 4, no. 4

SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

ISSN 2305-9052

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

South Ural State University

The scope of the journal:

- 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 computing
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents

УДК 004.657 DOI: 10.14529/cmse150401

КОЛОНОЧНЫЙ СОПРОЦЕССОР БАЗ ДАННЫХ ДЛЯ КЛАСТЕРНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Е.В. Иванова, Л.Б. Соколинский

Статья посвящена вопросам проектирования и реализации колоночного сопроцессора баз данных для реляционных СУБД. Колоночный сопроцессор (КСОП) разработан на базе колоночной модели хранения данных и ориентирован на большие кластерные вычислительные системы. КСОП может работать как на обычных центральных процессорах, так и на сопроцессорах с архитектурой MIC. КСОП поддерживает колоночные индексы с суррогатными ключами, которые во фрагментированном виде хранятся в оперативной памяти кластерной вычислительной системы. Фрагментация осуществляется на основе доменно-интервального принципа. На запросах класса OLAP колоночный сопроцессор КСОП демонстрирует производительность, многократно превышающую производительность строчных хранилищ.

Ключевые слова: колоночный сопроцессор, КСОП, распределенные колоночные индексы, доменно-интервальная фрагментация, кластерные вычислительные системы, многоядерные сопроцессоры, архитектура MIC.

Введение

В последнее время большую популярность приобрели колоночные СУБД (column-store DBMS) [1, 2]. В колоночной СУБД каждое отношение базы данных физически разбивается на столбцы (колонки), которые хранятся отдельно друг от друга. Раздельное хранение колонок позволяет колоночной СУБД считывать с диска только те колонки, которые соответствуют атрибутам, задействованным в запросе, в то время как строчная СУБД всегда считывает кортежи целиком. Это позволяет значительно экономить время на операциях ввода-вывода. Аналогичные преимущества колоночные СУБД получают при копировании данных из основной памяти в регистры. Помимо этого, колоночные СУБД позволяют эффективно использовать ряд технических приемов, недоступных или неэффективных для строчных СУБД. Первоначально колоночные хранилища были реализованы в ряде «академических» СУБД, среди которых следует упомянуть MonetDB [3], VectorWise (первоначальное название MonetDB/X100) [4] и C-Store [5]. Одной из первых коммерческих колоночных СУБД стала SybaseIQ [6]. Академические колоночные СУБД VectorWise и C-Store позднее эволюционировали в коммерческие системы Ingres VectorWise [7] и Vertica [8] соответственно. Начиная с 2013 года все основные производители СУБД включили в линейку продуктов колоночные версии своих систем: IBM [9], Microsoft [10–12], SAP [13] и Oracle [14]. В качестве современных колоночных СУБД также можно отметить EXASOL [15–17], Actian Vector [18], InfoBright [19] и SAND [20].

На рис. 1 схематично изображено основное отличие в физической организации колоночных и строчных хранилищ [2]: показаны три способа представления отношения Sales (Продажи), содержащего пять атрибутов. При колоночно-ориентированном подходе (рис. 1а и 1б) каждая колонка хранится независимо как отдельный объект базы данных. Поскольку данные в СУБД записываются и считываются поблочно, колоночно-ориентированный подход предполагает, что каждый блок, содержащий информацию из таблицы продаж, включает в себя данные только по некоторой единственной колонке.

Рис. 1. Физическая организация колоночных и строчных хранилищ

В этом случае, запрос, вычисляющий, например, число продаж определенного продукта за июль месяц, должен получить доступ только к колонкам prodid (идентификатор продукта) и date (дата продажи). Следовательно, СУБД достаточно загрузить в оперативную память только те блоки, которые содержат данные из этих колонок. С другой стороны, при строчно-ориентированном подходе (рис. 1в) существует единственный объект базы данных, содержащий все необходимые данные, то есть каждый блок на диске, содержащий информацию из таблицы Sales, включает в себя данные из всех колонок этой таблицы. В этом случае отсутствует возможность выборочно считать конкретные атрибуты, необходимые для конкретного запроса, без считывания всех остальных атрибутов отношения. Принимая во внимание тот факт, что затраты на обмены с диском (либо обмены между процессорным кэшем и оперативной памятью) являются узким место в системах баз данных, а схемы баз данных становятся все более сложными с включением широких таблиц с сотнями атрибутов, колоночные хранилища способны обеспечить существенный прирост в производительности при выполнении запросов класса OLAP.

Одним из главных преимуществом строчных хранилищ является наличие в строковых СУБД мощных процедур оптимизации запросов, разработанных на базе реляционной модели. Строковые СУБД также имеют большое преимущество в скорости обработки запросов класса OLTP. В соответствии с этим в исследовательском сообществе баз данных были предприняты интенсивные усилия по интеграции преимуществ столбцовой модели хранения данных в строковые СУБД [21]. Анализ предложенных решений показывает, что нельзя получить выгоду от хранения данных по столбцам, воспользовавшись системой баз данных со строковым хранением с вертикально разделенной схемой, либо проиндексировав все столбцы, чтобы обеспечить к ним независимый доступ.

Анализ современных тенденций в развитии аппаратного обеспечения и технологий баз данных говорит в пользу оптимальности следующего выбора среди возможных решений при разработке СУБД для обработки больших данных, схематично изображенного на рис. 2. Перспективным решением является создание колоночного сопроцессора КСОП (*CCOP — Columnar COProcessor*), совместимого с реляционной СУБД. Колоночный сопроцессор должен поддерживать распределенные *колоночные индексы*, постоянно хранимые в оперативной памяти кластерной вычислительной системы с многоядерными процессорными устройствами. Для взаимодействия с КСОП СУБД должна эмулировать материализационную модель обработки запроса. Суть материализационной модели состоит в том, что промежуточные отношения вычисляются полностью, за исключением атрибутов, не входящих в предикаты вышестоящих реляционных операций.

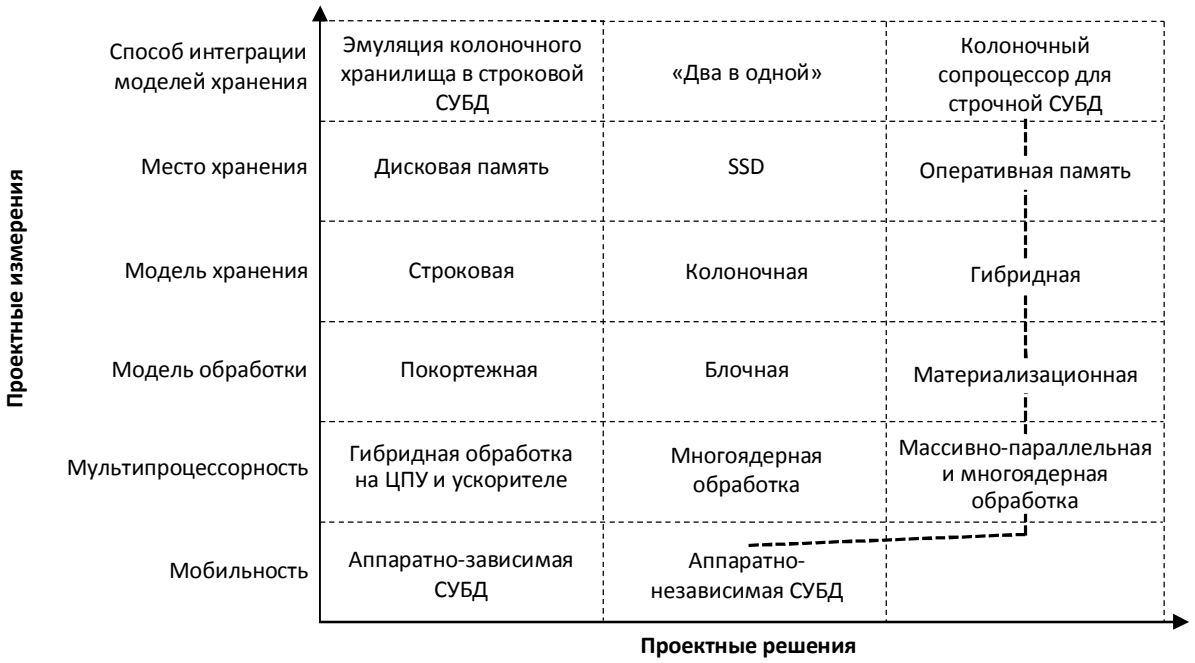


Рис. 2. Пространство проектных решений при разработке СУБД для больших данных

Это достигается переписыванием исходного SQL-запроса в последовательность запросов, использующих материализуемые представления. При таком подходе любая операция в плане выполнения запроса может быть заменена на вызов КСОП при условии, что в его памяти существуют необходимые колоночные индексы. Для сокращения расходов на разработку и сопровождение КСОП, он должен использовать аппаратно-независимые алгоритмы. Временные потери, связанные с отсутствием тонкого тюнинга под конкретную аппаратную платформу, должны компенсироваться хорошей масштабируемостью всех основных алгоритмов КСОП, используемых для выполнения запроса и для модификации колоночных индексов.

В данной, статье описываются архитектура, методы проектирования и реализации колоночного сопроцессора КСОП, удовлетворяющего вышеперечисленным требованиям. Теоретической основой КСОП является доменно-интервальная модель представления распределенных колоночных индексов, предложенная авторами в работах [22–25]. Статья имеет следующую структуру. В разделе 1 рассматривается архитектура колоночного сопроцессора КСОП, приводится общая схема взаимодействия СУБД и КСОП, даны интерфейсы основных операций, поддерживаемых КСОП. В разделе 2 описывается метод фрагментации и сегментации колоночных индексов, а также методы кодирования и сжатия данных, используемые в КСОП. В разделе 3 приведен пример выполнения запроса, поясняющий общую логику работы КСОП. В раздел 4 приводится структура КСОП и описываются схемы реализации основных операторов. В разделе 5 приводятся результаты вычислительных экспериментов по исследованию эффективности КСОП. В заключении суммируются полученные результаты, делаются итоговые выводы и обозначаются направления дальнейших исследований.

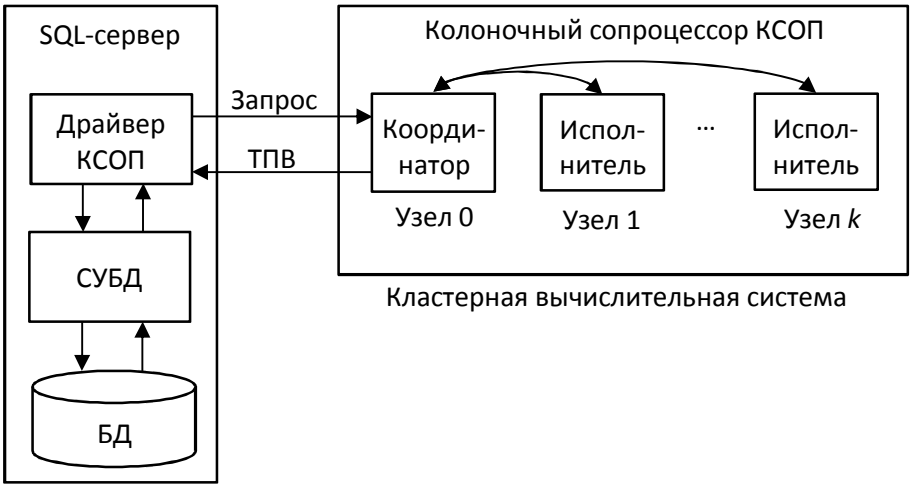


Рис. 3. Взаимодействие SQL-сервера с колоночным сопроцессором КСОП

1. Системная архитектура

Колоночный сопроцессор КСОП — это программная система, предназначенная для управления распределенными колоночными индексами, размещенными в оперативной памяти кластерной вычислительной системы. Назначение КСОП — вычислять таблицы предварительных вычислений для ресурсоемких реляционных операций по запросу СУБД. Общая схема взаимодействия СУБД и КСОП изображена на рис. 3. КСОП включает в себя программу «*Координатор*», запускаемую на узле вычислительного кластера с номером 0, и программу «*Исполнитель*», запускаемую на всех остальных узлах, выделенных для работы КСОП. На SQL-сервере устанавливается специальная программа «Драйвер КСОП», обеспечивающая взаимодействие с координатором КСОП по протоколу TCP/IP.

КСОП работает только с данными целых типов 32 или 64 байта. При создании колоночных индексов для атрибутов других типов, их значение кодируется в виде целого числа, или вектора целых чисел. В последнем случае длина вектора является фиксированной и называется *размерностью колоночного индекса*.

КСОП поддерживает следующие основные операции, доступные СУБД через интерфейс драйвера КСОП.

- CreateColumnIndex(TableID, ColumnID, SurrogateID, Width, Bottom, Top, Dimension) — создание распределенного колоночного индекса для атрибута ColumnID отношения TableID с параметрами: SurrogateID — идентификатор суррогатного ключа, Width — разрядность (32 или 64 бита); Bottom, Top — нижняя и верхняя границы доменного интервала; Dimension — размерность колоночного индекса. Возвращаемое значение: CIndexID — идентификатор созданного колоночного индекса.
- Insert(CIndexID, SurrogateKey, Value[*]) — добавление в колоночный индекс CIndexID нового кортежа (SurrogateKey, Value[*]).
- TransitiveInsert(CIndexID, SurrogateKey, Value[*], TValue[*]) — добавление в колоночный индекс CIndexID нового кортежа (SurrogateKey, Value[*]) с фрагментацией и сегментацией, определяемыми значением TValue[*].

- Delete(CIndexID, SurrogateKey, Value[*]) — удаление из колоночного индекса кортежа (SurrogateKey, Value[*]).
- TransitiveDelete(TCIndexID, SurrogateKey, TransitiveValue[*]) — удаление из колоночного индекса кортежа (SurrogateKey, Value[*]) с фрагментацией и сегментацией, определяемыми значением TValue[*].
- Execute(Query) — выполнение запроса на вычисление ТПВ с параметрами: Query — символьная строка, содержащая запрос в формате JSON. Примеры запросов приведены в разделе 3.2. Возвращаемое значение: PCTID — идентификатор Таблицы предварительных вычислений.

2. Управление данными

Все данные (колоночные индексы и метаданные), с которыми работает КСОП, хранятся в распределенной памяти кластерной вычислительной системы. Для колоночных индексов поддерживается двухуровневая система их разбиения на непересекающиеся части. Поясним ее работу на примере, изображенном на рис. 4, где показано разбиение колоночного индекса, построенного для атрибута B. В основе разбиения лежит домен $\mathfrak{D}_B = [0;99]$, на котором определен атрибут B. Домен разбивается на сегментные интервалы равной длины: [0; 11), [11; 22), ..., [77; 88), [88; 99]. Количество сегментных интервалов должно превышать суммарное количество процессорных ядер на узлах-исполнителях. Сегментные интервалы разбиваются на последовательные группы, которые называются *фрагментными интервалами*. В примере на рис. 4 это следующие интервалы: [0; 22), [22; 55), [55; 99]. Количество фрагментных интервалов должно совпадать с количеством узлов-исполнителей. Длины фрагментных интервалов могут не совпадать. Это необходимо для балансировки загрузки процессорных узлов в условиях перекоса данных.

На первом этапе распределения данных исходный колоночный индекс разбивается на фрагменты. Каждому фрагменту соответствует определенный фрагментный интервал. Кортеж (a, b) попадает в данный фрагмент тогда, и только тогда, когда значение b принадлежит соответствующему фрагментному интервалу. Все кортежи, принадлежащие одному фрагменту, хранятся на одном и том же процессорном узле. На втором этапе каждый фрагмент разбивается на сегменты. Каждому сегменту соответствует определенный сегментный интервал. Кортеж (a, b) попадает в данный сегмент тогда, и только тогда, когда значение b принадлежит соответствующему сегментному интервалу. Внутри каждого сегментного интервала записи сортируются в порядке возрастания значения атрибута.

В случае неравномерного распределения значений атрибута, по которому создан колоночный индекс, можно добиться примерно одинакового размера фрагментов, сдвигая границы фрагментных интервалов, как это сделано в примере на рис. 4. При этом могут получиться сегменты разной длины. В КСОП сегмент является наименьшей единицей распределения работ между процессорными ядрами. Если количество сегментов не велико по сравнению с количеством процессорных ядер на одном узле, то при выполнении запроса мы получим дисбаланс в загрузке процессорных ядер. Проблему балансировки

загрузки можно эффективно решить, уменьшив длину сегментных интервалов, и тем самым увеличив количество сегментов.

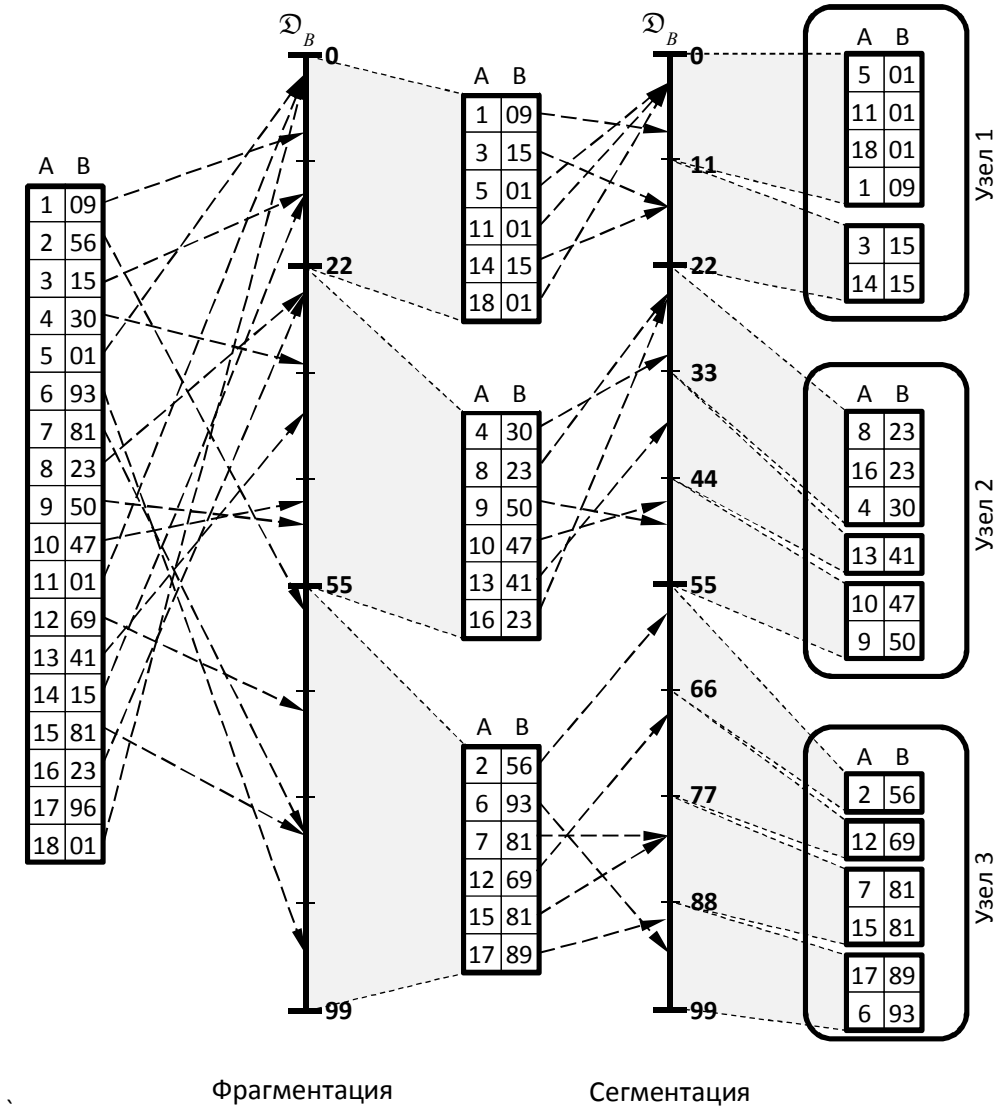


Рис. 4. Пример двухуровневого разбиения колоночного индекса на фрагменты и сегменты

КСОП работает только с данными целых типов (32 или 64 бита). Данные других типов должны быть предварительно закодированы драйвером КСОП в виде последовательности целых чисел. При этом могут использоваться методы, описанные в работе [11]. В случае, когда значение атрибута кодируется в виде целочисленного вектора размерности n (например, это может быть применено для длинных символьных строк), домен превращается в n-мерный куб. В этой ситуации n-мерный куб разбивается на n-мерные параллелепипеды по числу процессорных узлов (аналог фрагментного интервала), каждый из которых разбивается на еще меньшие n-мерные параллелепипеды по числу процессорных ядер в одном узле (аналог сегментного интервала).

Для сжатия закодированных сегментов могут использоваться как «тяжеловесные» методы (например, Хаффмана [26] или Лемпеля—Зива [27]), так и «легковесные»

(например, Run-Length Encoding [28, 29] или Null Suppression [30]), либо их комбинация [31]. В версии КСОП, описываемой в этой статье, для сжатия сегментов использовалась библиотека Zlib [32, 33], реализующая метод сжатия DEFLATE [34], являющийся комбинацией методов Хаффмана и Лемпеля—Зива. В работе [28] было показано, что легковесные методы типа Run-Length Encoding в случае колоночного представления информации могут оказаться более эффективными, чем тяжеловесные, поскольку допускают выполнение операций над данными без их распаковки.

3. Пример выполнения запроса

Поясним общую логику работы КСОП на простом примере. Пусть имеется база данных из двух отношений $R(A,B,D)$ и $S(A,B,C)$, хранящихся на SQL-сервере (см. рис. 5). Пусть нам необходимо выполнить запрос:

```
SELECT D, C
FROM R, S
WHERE R.B = S.B AND C<13.
```

Предположим, что КСОП имеет только два узла-исполнителя и на каждом узле имеется три процессорных ядра (процессорные ядра на рис. 5 промаркированы обозначениями P_{11},..., P_{23}). Положим, что атрибуты $R.B$ и $S.B$ определены на домене целых чисел из интервала [0; 120). Сегментные интервалы для $R.B$ и $S.B$ определим следующим образом: [0; 20), [20; 40), [40; 60), [60; 80), [80; 100), [100; 120). В качестве фрагментных интервалов для $R.B$ и $S.B$ зафиксируем: [0; 59] и [60; 119]. Пусть атрибут $S.C$ определен на домене целых чисел из интервала [0; 25]. Изначально администратор базы данных с помощью драйвера КСОП создает для атрибутов $R.B$ и $S.B$ распределенные колоночные индексы $I_{R.B}$ и $I_{S.B}$. Затем для атрибута $S.C$ создается распределенный колоночный индекс $I^B_{S.C}$, который фрагментируется и сегментируется транзитивно относительно индекса $I_{S.B}$. Распределенные колоночные индексы $I_{R.B}$, $I_{S.B}$ и $I^B_{S.C}$ сохраняются в оперативной памяти узлов-исполнителей. Таким образом мы получаем распределение данных внутри КСОП, приведенное на рис. 5. При поступлении SQL-запроса, приведенного выше, он преобразуется драйвером КСОП в план, определяемый следующим выражением реляционной алгебры:

$$\pi_{I_{R.B}.A \to A_R, I_{S.B}.A \to A_S} \left(I_{R.B} \underset{\boxed{\substack{I_{R.B}.B= \\ I_{S.B}.B}}}{\bowtie} \left(I_{S.B} \bowtie \sigma_{C<13} \left(I^B_{S.C} \right) \right) \right).$$

При выполнении драйвером операции Exec указанный запрос передается координатору КСОП в виде оператора CCOPQL в формате JSON. Запрос выполняется независимо процессорными ядрами узлов-исполнителей над соответствующими группами сегментов. При этом за счет доменной фрагментации и сегментации не требуются обмены данными как между узлами-исполнителями, так и между процессорными ядрами одного узла. Каждое процессорное ядро вычисляет свою часть ТПВ, которая пересылается на узел-координатор. Координатор объединяет фрагменты ТПВ в единую таблицу и пересылает ее драйверу, который выполняет материализацию этой таблицы в виде от-

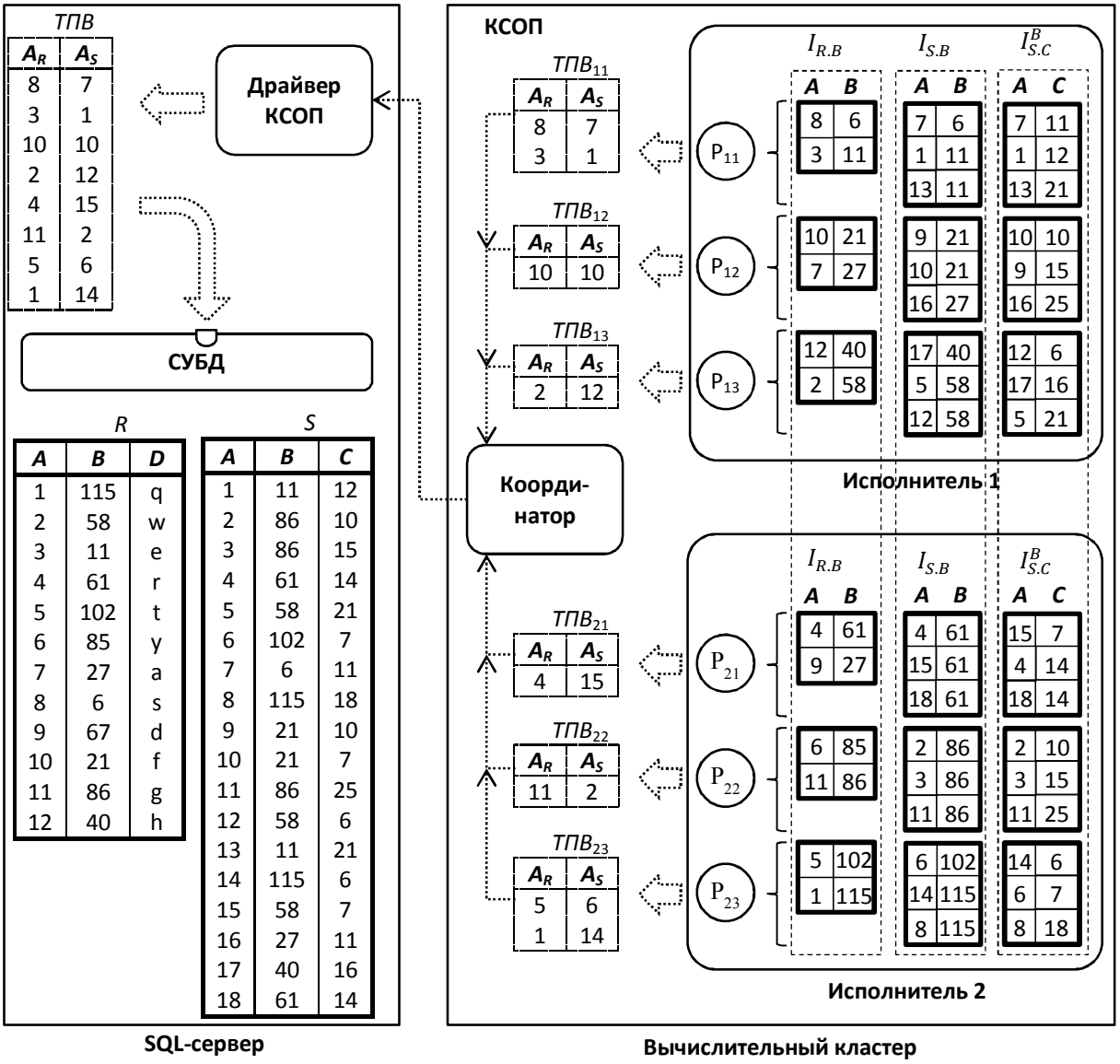


Рис. 5. Вычисление ТПВ с использованием КСОП

ношения в базе данных, хранящейся на SQL-сервере. После этого SQL-сервер вместо исходного SQL-оператора выполняет следующий оператор:

```
SELECT D, C
FROM
   R INNER JOIN (
      ТПВ INNER JOIN S ON (S.A = ТПВ.A_S)
   ) ON (R.A = ТПВ.A_R).
```

При этом используются обычные кластеризованные индексы в виде B-деревьев, заранее построенные для атрибутов $R.A$ и $S.A$. После выполнения запроса ТПВ удаляется, а распределенные колоночные индексы $I_{R.B}$, $I_{S.B}$ и $I^B_{S.C}$ сохраняются в оперативной памяти узлов-исполнителей для последующего использования.

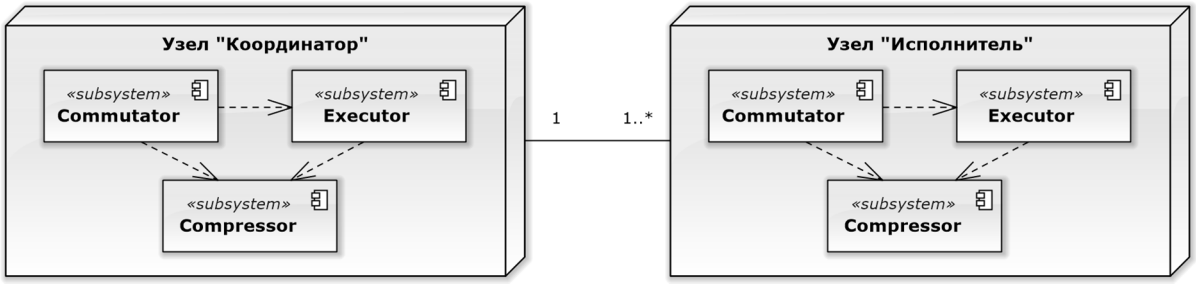


Рис. 6. Структура КСОП

4. Проектирование и реализация

На рис. 6 представлена диаграмма развертывания КСОП, основанная на нотации стандарта UML 2.0. КСОП является распределенной системой и включает в себя два типа узлов: координаторы и исполнители. В простейшем случае в системе имеется один узел-координатор и несколько узлов-исполнителей, как это показано на рис. 6. Однако, если при большом количестве исполнителей координатор становится узким местом, в системе может быть несколько координаторов. В этом случае они образуют иерархию в виде сбалансированного дерева, листьями которого являются узлы-исполнители.

И координатор, и исполнитель имеют унифицированную структуру, состоящую из трех компонент: *Commutator* (*Коммутатор*), *Executor* (*Исполнитель*), *Compressor* (*Модуль сжатия*). Однако их реализации в координаторе и исполнителе различается. Компонент Commutator в координаторе выполняет две функции: 1) обмен сообщениями в формате JSON с драйвером КСОП по протоколу TCP/IP (см. рис. 3); 2) обмен сообщениями с исполнителями по технологии MPI. Компонент Commutator в исполнителе осуществляет обмен сообщениями с координатором по технологии MPI. Компонент Executor на исполнителе организует параллельную обработку сегментов колоночных индексов, используя технологию OpenMP, и формирует фрагмент ТПВ. Компонент Executor на координаторе объединяет фрагменты ТПВ, вычисленные исполнителями в единую таблицу. Компоненты Commutator и Executor используют компонент Compressor для сжатия/распаковки данных. Рассмотрим, как КСОП выполняет основные операции над распределенными колоночными индексами, перечисленные в разделе 1.

При выполнении оператора *CreateColumnIndex* координатор добавляет информацию о структуре колоночного индекса в свой локальный словарь и отправляет исполнителям указание создать в своих локальных словарях дескриптор с информацией о новом колоночном индексе. При этом координатор определяет длину сегментного интервала и границы фрагментных интервалов. Эта информация сохраняется в дескрипторе колоночного индекса. Кроме этого, дескриптор включает в себя битовую шкалу сегментов, в которой значение 1 соответствует непустым сегментам, значение 0 — пустым. При начальном создании колоночного индекса все сегменты на всех узлах-исполнителях являются пустыми.

При выполнении оператора *Insert* координатору передается кортеж (SurrogateKey, Value) для вставки в указанный колоночный индекс. Координатор определяет, в границы какого фрагментного интервала попадает значение Value, и пересылает этот кортеж на соответствующий узел-исполнитель. Исполнитель определяет номер сегментного интервала, к которому принадлежит значение Value. Если соответствующий сегмент не

пуст, то кортеж вставляется в сегмент с сохранением упорядочения по полю Value. Если соответствующий сегмент пуст, то создается новый сегмент из одного кортежа.

При выполнении оператора *TransitiveInsert* координатору кроме нового кортежа (SurrogateKey, Value) передается дополнительное значение TValue, которое им используется для определения номера фрагментного интервала. Кортеж (SurrogateKey, Value) вместе со значением TValue пересылается на соответствующий узел-исполнитель. Исполнитель по значение TValue определяет номер сегментного интервала и вставляет новый кортеж в соответствующий сегмент.

При выполнении оператора *Delete* координатору передается суррогатный ключ SurrogateKey и значение Value, которые надо удалить. Координатор определяет, в границы какого фрагментного интервала попадает значение Value, и пересылает этот кортеж на соответствующий узел-исполнитель. Исполнитель определяет номер сегментного интервала, к которому принадлежит значение Value, производит в соответствующем сегменте поиск кортежа с ключом SurrogateKey и выполняет его удаление.

Оператор *TransitiveDelete* выполняется аналогично оператору *Delete* с той лишь разницей, что номера фрагментного и сегментного интервалов вычисляются по транзитивному значению TValue.

Выполнение оператора *Execute* включает в себя две фазы: 1) вычисление фрагментов ТПВ на узлах-исполнителях; 2) слияние фрагментов ТПВ в единую таблицу на узле-координаторе и пересылка ее на SQL-сервер. На первой фазе процессорные ядра (легковесные процессы OpenMP) выбирают необработанные сегментные интервалы и выполняют вычисление *сегментных* ТПВ для соответствующих сегментов колоночных индексов, задействованных в запросе. После того, как все сегментные интервалы обработаны, получившийся фрагмент ТПВ пересылается на узел-координатор в сжатом виде. На второй фазе координатор распаковывает все полученные сегментные ТПВ и объединяет их в единую таблицу. Для распараллеливания этого процесса используется технология OpenMP. Получившаяся ТПВ пересылается на SQL-сервер. При этом также может использоваться сжатие данных.

Колоночный сопроцессор КСОП был реализован на языке Си с использованием аппаратно независимых параллельных технологий MPI и OpenMP. Он может работать как на ЦПУ Intel Xeon X5680, так и на сопроцессоре Intel Xeon Phi без модификации кода. Объем исходного кода составил около двух с половиной тысяч строк. Исходные тексты КСОП свободно доступны в сети Интернет по адресу: https://github.com/elena-ivanova/colomnindices/.

5. Результаты экспериментов

В данном разделе приводятся результаты вычислительных экспериментов по исследованию эффективности колоночного сопроцессора КСОП.

5.1. Вычислительная среда

Эксперименты проводились на двух вычислительных комплексах с кластерной архитектурой: «Торнадо ЮУрГУ» Южно-Уральского государственного университета и «RSC PetaStream» Межведомственного суперкомпьютерного центра Российской академии наук. Основные параметры этих систем приведены в табл. 1.

Таблица 1

Параметры вычислительных комплексов

Параметры	**Вычислительный комплекс**	
	«Торнадо ЮУрГУ»	**«RSC PetaStream»**
Количество узлов	384	64
Тип процессоров	2 × Intel Xeon X5680 (12 ядер по 3.33 ГГц; 2 потока на ядро)	
Оперативная память узла	24 Гб	
Тип сопроцессора	Intel Xeon Phi SE10X (61 ядро по 1.1 ГГц; 4 потока на ядро)	Intel Xeon Phi 7120 (61 ядро по 1.24 ГГц)
Память сопроцессора	8 Гб	16 Гб
Тип системной сети	InfiniBand QDR (40 Гбит/с)	InfiniBand FDR (56 Гбит/с)
Тип управляющей сети	Gigabit Ethernet	Gigabit Ethernet
Операционная система	Linux CentOS 6.2	Linux CentOS 7.0

Для тестирования колоночного сопроцессора КСОП использовалась синтетическая база данных, построенная на основе эталонного теста TPC-H [35]. Тестовая база данных состояла из двух таблиц: **ORDERS** (ЗАКАЗЫ) и **CUSTOMER** (КЛИЕНТЫ), схема которых приведена на рис. 7. Для масштабирования базы данных использовался *масштабный коэффициент SF* (*Scale Factor*), значение которого изменялось от 1 до 10. При проведении экспериментов размер отношения **ORDERS** составлял SF×63 000 000 кортежей, размер отношения **CUSTOMER** — SF×630 000 кортежей.

Для генерации тестовой базы данных была написана программа на языке Си, использующая методику, изложенную в работе [36]. Атрибуты **CUSTOMER.A** и **ORDERS.A**, игравшие роль суррогатных ключей, заполнялись целыми числами с шагом 1, начиная со значения 0. Атрибут **CUSTOMER.ID_CUSTOMER**, являющийся первичным ключом, заполнялся целыми числами с шагом 1, начиная со значения 1. Атрибут **ORDERS.ID_ORDER**, являющийся первичным ключом, также заполнялся целыми числами с шагом 1, начиная со значения 1. Атрибут **ORDERS.ID_CUSTOMER**, являющийся внешним ключом, заполнялся значениями из интервала [1, SF*630000]. При этом, для имитирования перекоса данных использовались следующие распределения:

- равномерное (uniform);
- «45-20»;
- «65-20»;
- «80-20».

Для генерации неравномерного распределения была использована вероятностная модель. В соответствии с этой моделью коэффициент перекоса θ, $(0 \le \theta \le 1)$ задает распределение, при котором каждому целому значению из интервала [1, SF*630000] назначается некоторый весовой коэффициент $p_i (i = 1, \ldots, N)$, определяемый формулой

$$p_i = \frac{1}{i^{\theta} \cdot H_N^{(\theta)}}, \sum_{i=1}^{N} p_i = 1,$$

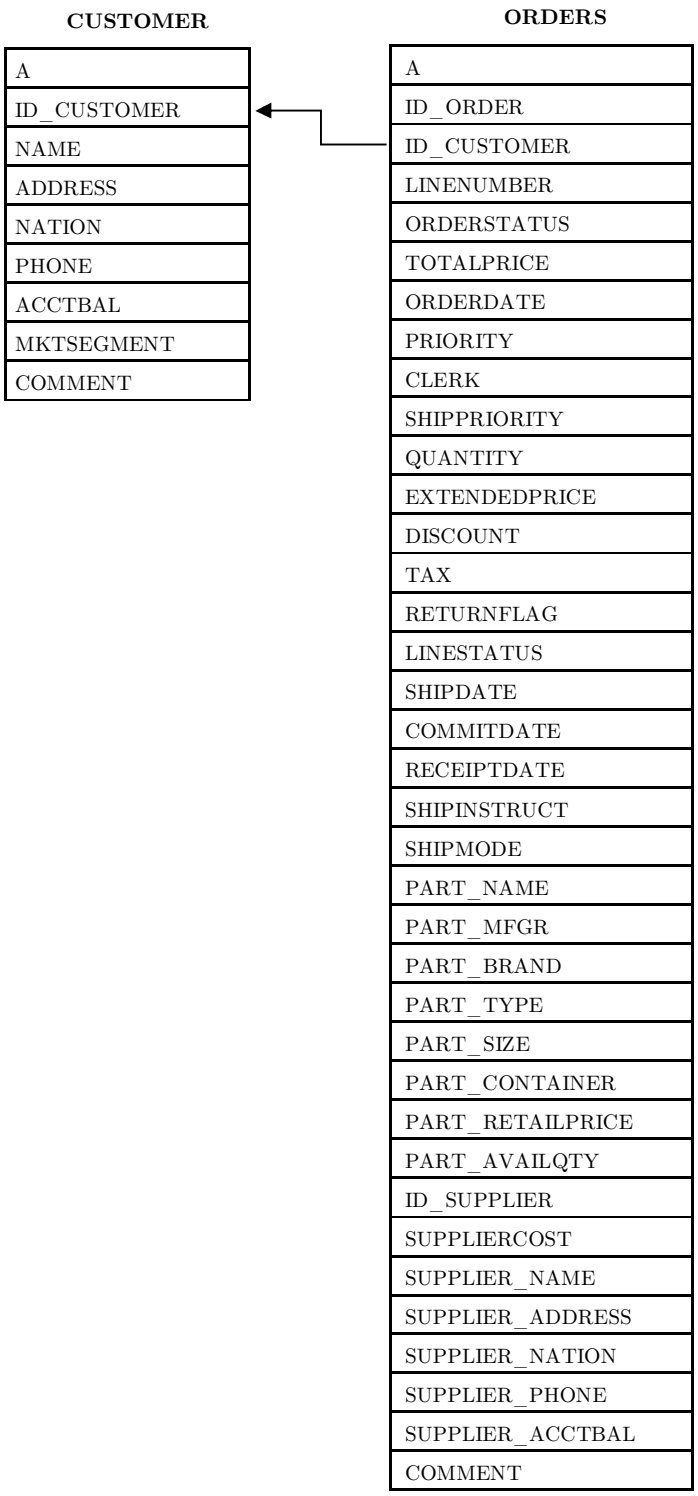


Рис. 7. Схема тестовой базы данных

где $N = \text{SF*}630000$ — количество различных значений атрибута **ORDERS.ID_CUSTOMER** и $H_N^{(s)} = 1^{-s} + 2^{-s} + \cdots + N^{-s}$, N-е гармоническое число порядка s. В случае $\theta = 0$ распределение весовых коэффициентов соответствует равномерному распределению. При $\theta = 0{,}86$ распределение соответствует правилу «80-20», в соответствии с которым 20 % самых популярных значений занимают 80 % позиций в столбце **ID_CUSTOMER** в таблице **ORDERS**. При $\theta = 0{,}73$ распределение соответ-

ствует правилу «65-20» (20 % самых популярных значений занимают 65 % позиций в столбце **ID_CUSTOMER** в таблице **ORDERS**). При $\theta = 0{,}5$ распределение соответствует правилу «45-20» (20 % самых популярных значений занимают 45 % позиций в столбце **ID_CUSTOMER** в таблице **ORDERS**).

Все остальные атрибуты в отношениях **CUSTOMER** и **ORDERS**, заполнялись случайными значениями соответствующих типов с равномерным распределением.

Тестовая база данных была развернута в СУБД PostgreSQL 9.4.0 на выделенном узле вычислительного кластера «Торнадо ЮУрГУ». В качестве тестового запроса фигурировал следующий SQL-запрос Q1:

```
# Запрос Q1
SELECT * FROM CUSTOMER, ORDERS
WHERE (CUSTOMER.ID_CUSTOMER=ORDERS.ID_CUSTOMER)
AND (ORDERS.TOTALPRICE <= Sel*100 000).
```

Для варьирования размера результирующего отношения использовался *коэффициент селективности Sel*, принимающий значение из интервала]0;1]. Коэффициент Sel определяет размер (в кортежах) результирующего отношения относительно размера отношения **ORDERS**. Например, при Sel = 0,5 размер результирующего отношения составляет 50 % от размера отношения **ORDERS**, при Sel = 0,05 — 5 %, а при Sel = 1 — 100 %. Таким образом, большая селективность запроса соответствует меньшему значению коэффициента селективности.

С помощью колоночного сопроцессора КСОП в оперативной памяти кластерной вычислительной системы были созданы следующие распределенные колоночные индексы:

$I_{CUSTOMER.ID\_CUSTOMER}(A,\ ID\_CUSTOMER)$,

$I_{ORDERS.ID\_CUSTOMER}(A,\ ID\_CUSTOMER)$,

$I_{ORDER.TOTALPRICE}(A,\ TOTALPRICE)$.

Все индексы сортировались по значениям соответствующих атрибутов. Индексы $I_{CUSTOMER.ID\_CUSTOMER}$, $I_{ORDERS.ID\_CUSTOMER}$ фрагментировались и сегментировались на основе доменно-интервального принципа по домену [1, SF*630000], индекс $I_{ORDER.TOTALPRICE}$ фрагментировался и сегментировался транзитивно относительно индекса $I_{ORDERS.ID\_CUSTOMER}$. Все фрагментные и сегментные интервалы, на которые делился домен, имели одинаковую длину. Сегменты индексов сжимались с помощью библиотеки Zlib.

При выполнении запроса колоночный сопроцессор КСОП вычислял таблицу предварительных вычислений $P(A\_ORDERS,\ A\_CUSTOMER)$ следующим образом:

$$P = \pi_{I_{CUSTOMER.ID\_CUSTOMER}.A \to A\_CUSTOMER,\ I_{ORDERS.ID\_CUSTOMER}.A \to A\_ORDERS}\Big(I_{CUSTOMER.ID\_CUSTOMER} \underset{\boxed{\substack{I_{CUSTOMER.ID\_CUSTOMER}.ID\_CUSTOMER = \\ I_{ORDERS.ID\_CUSTOMER}.ID\_CUSTOMER}}}{\bowtie} \left(I_{ORDERS.ID\_CUSTOMER} \bowtie \sigma_{TOTALPRICE \le Sel \cdot 100\,000}\left(I_{ORDERS.TOTALPRICE}\right)\right)\Big).$$

Вычисления фрагментов таблицы P производились параллельно на доступных процессорных узлах без обменов данными. Затем полученные фрагменты пересылались на узел с PostgreSQL, где *координатор* осуществлял их слияние в общую таблицу P. Вместо выполнения запроса Q1 в PostgreSQL выполнялся следующий запрос Q2 с использованием таблицы предварительных вычислений P:

```
# Запрос Q2
SELECT * FROM
   CUSTOMER INNER JOIN (
      P INNER JOIN ORDERS ON (ORDERS.A = P.A_ORDERS)
   ) ON (CUSTOMER.A=P.A_CUSTOMER);
```

При этом в PostgreSQL для атрибутов CUSTOMER.A и ORDERS.A предварительно были созданы кластеризованные индексы в виде B-деревьев.

5.2. Балансировка загрузки процессорных ядер Xeon Phi

В разделе 2 было сказано, что балансировку загрузки процессорных узлов, на которых располагаются распределенные колоночные индексы, можно осуществлять путем сдвигов границ фрагментных интервалов соответствующих доменов. В отличие от этого все сегментные интервалы для каждого конкретного домена имеют одинаковую длину. В случае, если значения атрибута в столбце распределены неравномерно, сегменты соответствующего колоночного индекса внутри одного процессорного узла могут значительно отличаться по своим размерам. Потенциально это может привести к дисбалансу в загрузке процессорных ядер, так как каждый сегмент обрабатывается целиком на одном ядре. Однако, если число обрабатываемых сегментов значительно превышает количество процессорных ядер, дисбаланса загрузки не возникнет. Целью первого эксперимента было определить оптимальное соотношение между количеством сегментов и процессорных ядер в условиях перекоса в распределении данных. Использовались следующие параметры эксперимента:

- Вычислительная система: «Торнадо ЮУрГУ»;
- Количество задействованных узлов: 1;
- Используемый процессор: Xeon Phi;
- Количество задействованных ядер: 60;
- Количество нитей на ядро: 1;
- Коэффициент масштабирования базы данных: SF = 1;
- Коэффициент селективности: Sel=0,0005;
- Распределение значений в колонке ORDERS.ID_CUSTOMER: uniform, «45-20», «65-20», 80-20»;
- Операция: построение таблицы предварительных вычислений P.

Результаты эксперимента представлены на рис. 8. Из графиков видно, что при малом количестве сегментов сильный перекос по данным приводит к существенному дисбалансу в загрузке процессорных ядер. В случае, когда количество сегментов равно 60 и совпадает с количеством ядер, время выполнения операции при распределении «80-20» более чем в четыре раза превышает время выполнения той же операции при равномерном (uniform) распределении. Однако при увеличении количества сегментов влияние перекоса по данным нивелируется. Для распределения «45-20» оптимальным оказывается число сегментов, равное 10 000, для распределения «65-20» — 20 000, и для распределения «80-20» — 200 000.

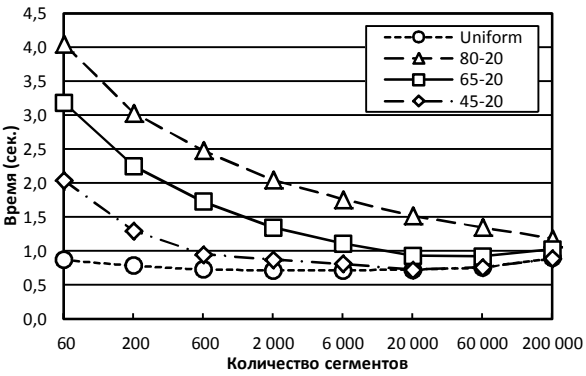


Рис. 8. Балансировка загрузки процессорных ядер сопроцессора Xeon Phi

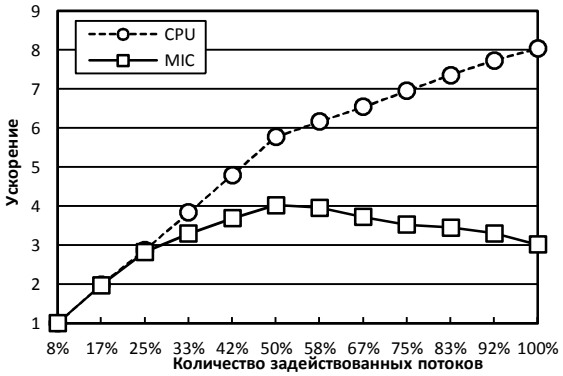


Рис. 9. Влияние гиперпоточности на ускорение

5.3. Влияние гиперпоточности

В большинстве случаев современные кластерные вычислительные системы оснащаются процессорами, поддерживающими технологию гиперпоточности (hyper-threading) [37], при включении которой каждое физическое ядро процессора определяется операционной системой как два или более логических ядра. У каждого логического ядра имеется свой набор регистров и контроллер прерываний. Остальные элементы физического ядра являются общими для всех логических ядер. При определенных рабочих нагрузках использование гиперпоточности позволяет увеличить производительность процессора.

ЦПУ Intel Xeon X5680 имеет аппаратную поддержку двух потоков на ядро, сопроцессор Intel Xeon Phi поддерживает четыре потока на ядро. Целью второго эксперимента было определить насколько эффективно гиперпоточность может быть применена при работе колоночного сопроцессора КСОП. Использовались следующие параметры эксперимента:

- Вычислительная система: «Торнадо ЮУрГУ»;
- Количество задействованных узлов: 1;
- Используемые процессорные устройства (ПУ):
  2 × Intel Xeon X5680 (CPU) / Intel Xeon Phi (MIC);
- Количество задействованных ядер: 12 / 60;
- Коэффициент масштабирования базы данных: SF = 1;
- Коэффициент селективности: Sel=0,0005;
- Распределение значений в колонке ORDERS.ID_CUSTOMER: uniform;
- Операция: построение таблицы предварительных вычислений P.

Эксперимент сначала проводился на 2 × Intel Xeon X5680 (CPU), а затем на Intel Xeon Phi (MIC). Варьировался процент задействованных потоков. Для 2 × Intel Xeon X5680 величина 100 % соответствует 24 потокам. Для Intel Xeon Phi величина 100 % соответствует 240 потокам. Ускорение вычислялось по формуле $t_{x\%}/t_{20\%}$, где $t_{x\%}$ — время выполнения операции по вычислению ТПВ на x % потоков, $t_{20\%}$ — на 20 % потоков. Результаты эксперимента представлены на рис. 9. Эксперимент показал, что для CPU производительность растет вплоть до максимального числа аппаратно поддерживаемых потоков. Однако, при использовании одного потока на ядро на CPU наблюдается ускорение, близкое к идеальному, а при использовании двух потоков на ядро прирост уско-

рения становиться более медленным. Применительно к MIC картина меняется. При использовании одного потока на ядро на MIC наблюдается ускорение, близкое к идеальному. При использовании двух потоков на ядро прирост ускорения становится более медленным. Использование же большего количества потоков на одно ядро ведет к деградации производительности.

5.4. Масштабируемость КСОП

В данном разделе исследуется масштабируемость колоночного сопроцессора КСОП на двух различных вычислительных системах и на базах данных двух разных масштабов. Масштабируемость является одной из важнейших характеристик при разработке параллельных СУБД для вычислительных систем с массовым параллелизмом. Основной численной характеристикой масштабируемости является ускорение, которое вычисляется по формуле t_x/t_k, где t_k — время выполнения запроса на некоторой базовой конфигурации, включающей k процессорных узлов, t_x — время выполнения запроса на конфигурации, включающей x процессорных узлов при $x \geq k$. При использовании колоночного сопроцессора КСОП общее время выполнения запроса вычисляется по следующей формуле:

$$t = t_{openMP} + t_{MPI} + t_{merge} + t_{SQL},$$

где t_{openMP} — максимальное время вычисления фрагментов ТПВ и их сжатие на отдельных процессорных узлах; t_{MPI} — время пересылки сжатых фрагментов ТПВ на узел-координатор; t_{merge} — время их распаковки и слияния в единую таблицу; t_{SQL} — время выполнения запроса с использованием ТПВ на SQL-сервере. В описываемой реализации роль SQL-сервера играла СУБД PostgeSQL, установленная на узле-координаторе. Поэтому время t_{SQL} не зависело от количества используемых процессорных узлов. Это время будет исследовано в разделе 5.5. В данном разделе мы исследуем время, вычисляемое по формуле:

$$t_{total} = t_{openMP} + t_{MPI} + t_{merge}.$$

Первый эксперимент на масштабируемость проводился на кластере «Торнадо ЮУрГУ» и имел следующие параметры:

- Вычислительная система: «Торнадо ЮУрГУ»;
- Количество задействованных узлов: 60–210;
- Используемые процессорные устройства (ПУ): 2 × Intel Xeon X5680;
- Использование гиперпоточности: не используется;
- Коэффициент масштабирования базы данных: SF = 1;
- Коэффициент селективности: 0,05–0,0005;
- Распределение значений в колонке ORDERS.ID_CUSTOMER: uniform;
- Операция: построение таблицы предварительных вычислений P.

Результаты эксперимента приведены на рис. 10. Графики ускорения вычисления ТПВ на рис. 10 показывают, что селективность запроса Sel оказывается фактором, ограничивающим масштабируемость. Так, при $Sel = 0{,}05$ масштабируемость ограничена 150 вычислительными узлами, при $Sel = 0{,}005$ — 180, а при $Sel = 0{,}0005$ превышает 210.

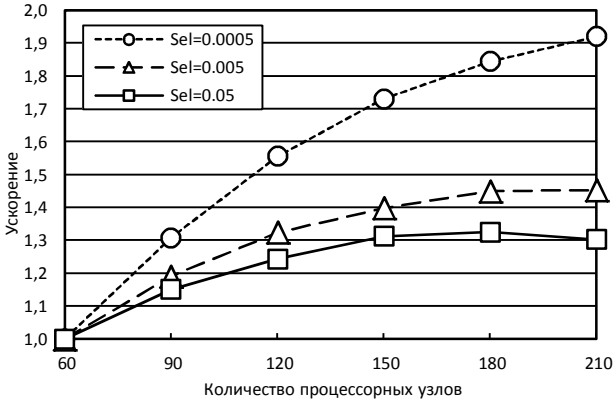


Рис. 10. Ускорение вычисления ТПВ на кластере «Торнадо ЮУрГУ» при SF = 1

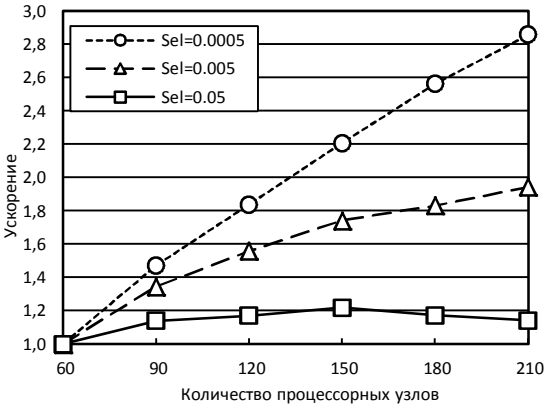
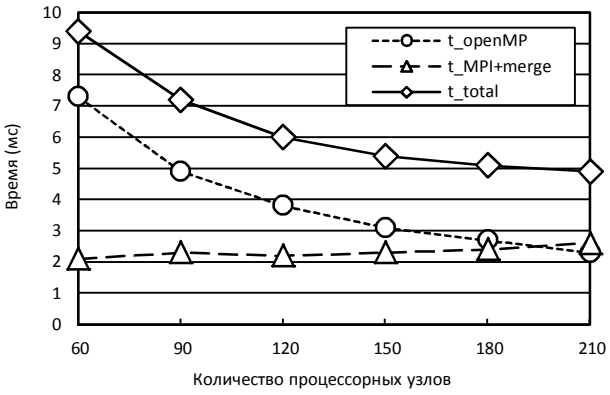
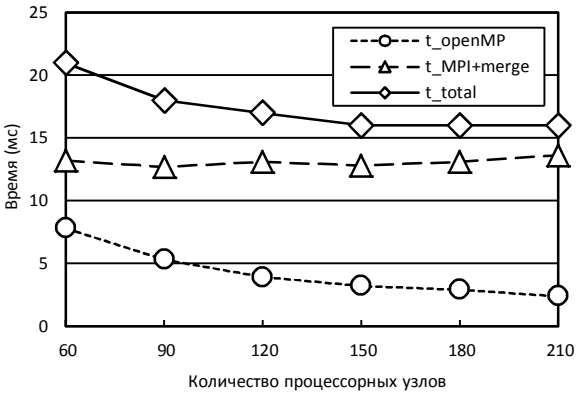


Рис. 11. Ускорение вычисления ТПВ на кластере «Торнадо ЮУрГУ» при SF = 10

Это связано с тем, что при увеличении селективности увеличивается размер ТПВ. При $Sel = 0.0005$ он составляет порядка 31 500 кортежей, при $Sel = 0.005$ — 315 000, а при $Sel = 0{,}05$ — 3 150 000 (значения даны для масштабного коэффициента SF = 1). На рис. 12 изображены зависимости выполнения времени подопераций от количества процессорных узлов. При малой селективности (рис. 12а) сумма времени пересылки фрагментов ТПВ, их сжатия, распаковки и слияния (t_MPI+merge) остается практически неизменной при увеличении процессорных узлов. Время вычисления фрагментов ТПВ на процессорных узлах (t_openMP) с ростом количества узлов уменьшается, и вплоть то 210 узлов превышает t_MPI+merge. Поэтому общее время (t_total) также уменьшается, что обеспечивает положительное ускорение. Однако при большой селективности (рис. 12б) t_MPI+merge, оставаясь практически неизменным, значительно превышает t_openMP, что препятствует существенному уменьшению общего времени при увеличении количества процессорных узлов.



а) Sel = 0,0005



б) Sel = 0,05

Рис. 12. Время выполнения подопераций при вычислении ТПВ на «Торнадо ЮУрГУ» при SF = 1

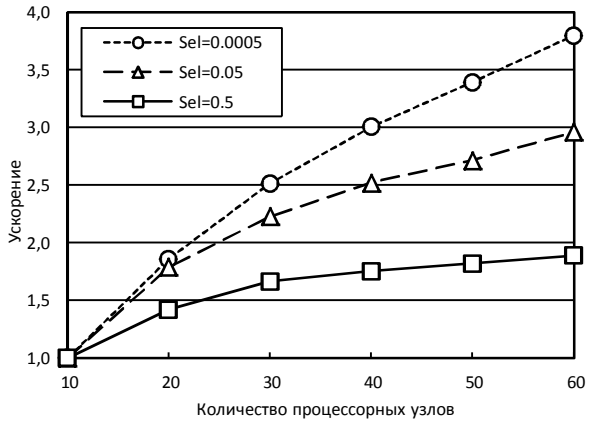


Рис. 13. Ускорение вычисления ТПВ на кластере «RSC PetaStream» при SF = 1

При увеличение коэффициента масштабируемости базы данных SF до значения 10 картина существенно меняется (см. рис. 11). На рис. 11 видно, что в этом случае кривая ускорения для $Sel = 0{,}0005$ становится практически линейной, а для $Sel = 0{,}005$ приближается к линейной. Однако, и в этом случае при большом значении коэффициента селективности $Sel = 0{,}05$ масштабируемость ограничивается 150 процессорными узлами. Причина та же самая: при $Sel = 0{,}05$ время передачи, распаковки и слияния ТПВ ($t_{MPI} + t_{merge}$) меняется мало и существенно превышает время ее вычисления t_{openMP}, в то время как при $Sel = 0{,}0005$ значение ($t_{MPI} + t_{merge}$) почти на порядок меньше t_{openMP}. В соответствие с этим можно сделать вывод, что при малой селективности запроса КСОП демонстрирует на больших базах данных ускорение, близкое к линейному.

При исследовании масштабируемости колоночного сопроцессора КСОП важным вопросом является ее зависимость от аппаратной архитектуры вычислительной системы. Как было сказано разделе 4, КСОП реализован с использованием аппаратно независимых параллельных технологий MPI и OpenMP. Поэтому он может работать как на ЦПУ Intel Xeon X5680, так и на сопроцессоре Intel Xeon Phi. Масштабируемость КСОП на вычислительном кластере «RSC PetaStream» была исследована в эксперименте, который имел следующие параметры:

- Вычислительная система: «RSC PetaStream»;
- Количество задействованных узлов: 10–60;
- Используемые процессорные устройства (ПУ): Intel Xeon Phi 7120;
- Использование гиперпоточности: не используется;
- Коэффициент масштабирования базы данных: SF = 1;
- Коэффициент селективности: 0,5–0,0005;
- Распределение значений в колонке ORDERS.ID_CUSTOMER: uniform;
- Операция: построение таблицы предварительных вычислений P.

Результаты эксперимента представлены на рис. 13. Они показывают, что масштабируемость КСОП и в этом случае приближается к линейной при уменьшении селективности запроса.

5.5. Использование КСОП при выполнении SQL-запросов

В заключительном эксперименте было исследовано в какой мере использование колоночного сопроцессора КСОП может ускорить выполнение запроса класса OLAP в реляционной СУБД. Эксперимент проводился при следующих параметрах:

- Вычислительная система: «Торнадо ЮУрГУ»;
- Конфигурация PostgreSQL: 1 процессорный узел с SSD Intel;
- Конфигурация КСОП: 210 процессорных узлов;
- Используемые процессорные устройства (ПУ): 2 × Intel Xeon X5680;
- Использование гиперпоточности: не используется;
- Коэффициент масштабирования базы данных: SF = 1;
- Коэффициент селективности: 0,05–0,0005;
- Распределение значений в колонке ORDERS.ID_CUSTOMER: uniform;
- Операция: выполнение SQL-запроса.

В ходе выполнения эксперимента исследовались следующие три конфигурации (см. табл. 2):

- PostgreSQL: выполнение запроса Q1 (см. раздел 5.1) без создания индексных файлов в виде B-деревьев;
- PostgreSQL & B-Trees: выполнение запроса Q1 (см. раздел 5.1) с предварительным созданием индексных файлов в виде B-деревьев для атрибутов CUSTOMER.ID_CUSTOMER и ORDERS.ID_CUSTOMER;
- PostgreSQL & CCOP: выполнение запроса Q2 (см. раздел 5.1) с использованием ТПВ P и предварительным созданием индексных файлов в виде B-деревьев для атрибутов CUSTOMER.A и ORDERS.A.

В последнем случае ко времени выполнения запроса добавлялось время создания ТПВ колоночным сопроцессором КСОП (CCOP). В каждом случае замерялось время первого и повторного запуска запроса. Это связано с тем, что после первого выполнения запроса PostgreSQL собирает статистическую информацию, сохраняемую в словаре базы данных, которая затем используется для оптимизации плана выполнения запроса.

Таблица 2

Время вычисления SQL-запроса и ускорение в сравнении с PostgreSQL при SF = 1

Конфигурация	Время (мин)					
	Sel = 0,0005		Sel = 0,005		Sel = 0,05	
	1-й запуск	2-й запуск	1-й запуск	2-й запуск	1-й запуск	2-й запуск
PostgreSQL	7,3	1,21	7,6	1,29	7,6	1,57
PostgreSQL & B-Trees	2,62	2,34	2,83	2,51	2,83	2,63
PostgreSQL & CCOP	0,073	0,008	0,65	0,05	2,03	1,72
Ускорение						
$\frac{t_{PostgreSQL}}{t_{PostgreSQL\ \&\ CCOP}}$	100	151	12	27	4	0,9
$\frac{t_{PostgreSQL\&B-Trees}}{t_{PostgreSQL\ \&\ CCOP}}$	36	293	4	50	1,4	1,53

Эксперименты показали, что при отсутствии индексов в виде B-деревьев использование колоночного сопроцессора позволяет увеличить производительность выполнения запроса в 100–150 раз для коэффициента селективности Sel = 0,0005. Однако при больших значениях коэффициента селективности эффективность использования КСОП может снижаться вплоть до отрицательных значений (ускорение меньше единицы).

Заключение

В статье описана общая архитектура организации системы баз данных с использованием колоночного сопроцессора КСОП. В состав системы входит SQL-сервер и вычислительный кластер. На SQL-сервере устанавливается реляционная СУБД с внедренным коннектором и драйвер КСОП. На вычислительном кластере устанавливается колоночный сопроцессор КСОП. Все колоночные индексы создаются и поддерживаются в распределенной оперативной памяти вычислительного кластера. При создании информационных систем с такой конфигурацией необходимо будет еще подключить подсистему восстановления колоночных индексов после сбоя. Для этого копии колоночных индексов и метаданных могут создаваться на твердотельных дисках, установленных на каждом вычислительном узле. Приведены результаты экспериментов над синтетической базой данных большого размера, исследующие эффективность колоночного сопроцессора КСОП. Эксперименты показали, что подходы и методы параллельного выполнения запросов класса OLAP на базе доменно-колоночной модели, демонстрируют хорошую масштабируемость (до нескольких сотен процессорных узлов и десятков тысяч процессорных ядер) для запросов с большой селективностью, которые являются типичными для хранилищ данных. Использование колоночного сопроцессора КСОП во взаимодействии с СУБД PostgreSQL позволило повысить эффективность выполнения таких запросов более чем на два порядка. Однако, эффективность использования КСОП снижается при уменьшении размеров базы данных и при увеличении размеров результирующего отношения.

В качестве направлений дальнейших исследований можно выделить следующие.

- Разработка и исследование методов интеграции КСОП со свободно распространяемыми реляционными СУБД типа PostgreSQL.
- Интеграция в КСОП легковесных методов сжатия, не требующих распаковки для выполнения операций над ними.
- Обобщение описанных подходов и методов на многомерные данные.

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

Литература

1. Чернышев, Г.А. Организация физического уровня колоночных СУБД / Г.А Чернышев // Труды СПИИРАН. — 2013. — № 7. Вып. 30. — С. 204–222.
2. Abadi, D.J. The Design and Implementation of Modern Column-Oriented Database Systems / D.J. Abadi, P.A. Boncz, S. Harizopoulos, S. Idreos, S. Madden // Foundations

and Trends in Databases. — 2013. — Vol. 5, No. 3. — P. 197–280. DOI: 10.1561/1900000024.

3. Idreos, S. MonetDB: Two Decades of Research in Column-oriented Database Architectures / S. Idreos, F. Groffen, N. Nes, S. Manegold, S. Mullender, M.L. Kersten // IEEE Data Engineering Bulletin. — 2012. — Vol. 35, No. 1. — P. 40–45.
4. Boncz, P.A. MonetDB/X100: Hyper-pipelining query execution / P.A. Boncz, M. Zukowski, N. Nes // Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR), January 4–7, Asilomar, CA, USA. — 2005. — P. 225–237.
5. Stonebraker, M. C-Store: A Column-Oriented DBMS / M. Stonebraker, D.J. Abadi, A. Batkin, X. Chen, M. Cherniack, M. Ferreira, E. Lau, A. Lin, S.R. Madden, E.J. O'Neil, P.E. O'Neil, A. Rasin, N. Tran, S.B. Zdonik // Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05), August 30 – September 2, 2005, Trondheim, Norway. — ACM, 2005. — P. 553–564.
6. MacNicol, R. Sybase IQ multiplex — designed for analytics / R. MacNicol, B. French // Proceedings of the Thirtieth International Conference on Very Large Data Bases, August 31–September 3, 2004, Toronto, Canada. — Morgan Kaufmann, 2004. — P. 1227–1230. DOI: 10.1016/b978-012088469-8/50111-x.
7. Zukowski, M. Vectorwise: Beyond column stores / M. Zukowski, P.A. Boncz // IEEE Data Engineering Bulletin. — 2012. — Vol. 35, No. 1. — P. 21–27.
8. Lamb, A. The Vertica analytic database: C-store 7 years later / A. Lamb, M. Fuller, R. Varadarajan, N. Tran, B. Vandier, L. Doshi, C. Bear // Proceedings of the VLDB Endowment. — 2012. — Vol. 5, No. 12. — P. 1790–1801. DOI: 10.14778/2367502.2367518.
9. Barber, R. Business Analytics in (a) Blink / R. Barber, P. Bendel, M. Czech, O. Draese, F. Ho, N. Hrle, S. Idreos, M.-S. Kim, O. Koeth, J.-G. Lee, T.T. Li, G.M. Lohman, K. Morfonios, R. Müller, K. Murthy, I. Pandis, L. Qiao, V. Raman, R. Sidle, K. Stolze, S. Szabo // IEEE Data Engineering Bulletin. — 2012. — Vol. 35, No. 1. — P. 9–14.
10. Larson, P.-A. Enhancements to SQL server column stores / P.-A. Larson, C. Clinciu, C. Fraser, E.N. Hanson, M. Mokhtar, M. Nowakiewicz, V. Papadimos, S.L. Price, S. Rangarajan, R. Rusanu, M. Saubhasik // Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD'13), June 22–27, 2013, New York, NY, USA. — ACM, 2013. — P. 1159–1168. DOI: 10.1145/2463676.2463708.
11. Larson, P.-A. SQL server column store indexes / P.-A. Larson, C. Clinciu, E.N. Hanson, A. Oks, S.L. Price, S. Rangarajan, A. Surna, Q. Zhou // Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (SIGMOD'11), June 12–16, 2011, Athens, Greece. — ACM, 2011. — P. 1177–1184. DOI: 10.1145/1989323.1989448.
12. Larson, P.-A. Columnar Storage in SQL Server 2012 / P.-A. Larson, E.N. Hanson, S.L. Price // IEEE Data Engineering Bulletin. — 2012. — Vol. 35, No. 1. — P. 15–20.
13. Färber, F. The SAP HANA Database — An Architecture Overview / F. Färber, N. May, W. Lehner, P. Große, I. Müller, H. Rauhe, J. Dees // IEEE Data Engineering Bulletin. — 2012. — Vol. 35, No. 1. — P. 28–33.
14. Weiss, R. A Technical Overview of the Oracle Exadata Database Machine and Exadata Storage Server — White Paper. — Oracle Corporation. — 2012. — 35 p. / R.A. Weiss. URL: http://www.oracle.com/technetwork/database/exadata/exadata-technical-whitepaper-134575.pdf (дата обращения: 29.10.2015).

15. A Drill-Down into EXASolution. — Technical Whitepaper. — EXASOL AG. — 2014. — 15 p. URL: http://info.exasol.com/whitepaper-exasolution-2-en.html (дата обращения: 22.10.2015).
16. A Peek under the Hood. — Technical Whitepaper. — EXASOL AG. — 2014. — 16 p. URL: http://www.breos.com/sites/default/files/pdf/downloads/exasol_whitepaper.pdf (дата обращения: 22.10.2015).
17. EXASolution. — Business Whitepaper. — EXASOL AG. — 2015. — 11 p. URL: http://info.exasol.com/business-whitepaper-exasolution-en.html (дата обращения: 27.10.2015).
18. Actian SQL Analytics in Hadoop. — A Technical Overview. — Actian Corporation. — 2015. — 16 p. URL: http://bigdata.actian.com/SQLAnalyticsinHadoop (дата обращения: 27.10.2015).
19. Ślęzak, D. Towards approximate SQL: infobright's approach / D. Ślęzak, M. Kowalski // Proceedings of the 7th international conference on Rough sets and current trends in computing (RSCTC'10). — Springer-Verlag, 2010. — P. 630–639. DOI: 10.1007/978-3-642-13529-3_67.
20. SAND CDBMS: A Technological Overview. — White Paper. — SAND Technology. — 2010. — 16 p. URL: http://www.sand.com/downloads/side2239eadd/wp_sand_cdbms_technological_overview_en.pdf (дата обращения: 29.10.2015).
21. Abadi, D.J. Column-Stores vs. Row-Stores: How Different Are They Really? / D.J. Abadi, S.R. Madden, N. Hachem // Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 9–12, 2008, Vancouver, BC, Canada. — ACM, 2008. — P. 967–980. DOI: 10.1145/1376616.1376712.
22. Ivanova, E. Decomposition of Natural Join Based on Domain-Interval Fragmented Column Indices / E. Ivanova, L. Sokolinsky // Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO, May 25–29, 2015, Opatija, Croatia. — IEEE, 2015. — P. 223–226. DOI: 10.1109/mipro.2015.7160266.
23. Иванова, Е.В. Декомпозиция операции группировки на базе распределенных колоночных индексов / Е.В. Иванова, Л.Б. Соколинский // Наука ЮУрГУ. — Челябинск: Издательский центр ЮУрГУ, 2015. — С. 15–23.
24. Иванова, Е.В. Декомпозиция операций пересечения и соединения на основе доменно-интервальной фрагментации колоночных индексов / Е.В. Иванова, Л.Б. Соколинский // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. — 2015. — Т. 4, № 1. — С. 44–56. DOI: 10.14529/cmse150104.
25. Иванова, Е.В. Использование распределенных колоночных хеш-индексов для обработки запросов к сверхбольшим базам данных / Е.В. Иванова // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: Труды Международной суперкомпьютерной конференции (22–27 сентября 2014 г., Новороссийск). — М.: Изд-во МГУ, 2014. — С. 102–104.
26. Huffman, D. A method for the construction of minimum-redundancy codes / D. Huffman // Proceedings of the I.R.E. — 1952. — Vol. 40, No. 9. — P. 1098–1101. DOI: 10.1109/jrproc.1952.273898.
27. Ziv, J. A universal algorithm for sequential data compression / J. Ziv, A. Lempel // IEEE Transactions on Information Theory. — 1977. — Vol. 23, No. 3. — P. 337–343. DOI: 10.1109/tit.1977.1055714.

28. Abadi, D.J. Integrating compression and execution in column-oriented database systems / D.J. Abadi, S.R. Madden, M. Ferreira // Proceedings of the 2006 ACM SIGMOD international conference on Management of data, June 26–29, 2006, Chicago, Illinois. — ACM, 2006. — P. 671–682. DOI: 10.1145/1142473.1142548.
29. Bassiouni, M.A. Data Compression in Scientific and Statistical Databases / M.A. Bassiouni // IEEE Transactions on Software Engineering. — 1985. — Vol. 11, No. 10. — P. 1047–1058. DOI: 10.1109/tse.1985.231852.
30. Ruth, S.S. Data Compression for Large Business Files / S.S. Ruth, P.J. Kreutzer // Datamation. — 1972. — Vol. 19, No. 9. — P. 62–66.
31. Roth, M.A. Database compression / M.A. Roth, S.J. Van Horn // ACM SIGMOD Record. — 1993. — Vol. 22, No. 3. — P. 31–39. DOI: 10.1145/163090.163096.
32. Deutsch, P. ZLIB Compressed Data Format Specification version 3.3 / P. Deutsch, J.-L. Gailly. — United States: RFC Editor. — 1996. DOI: 10.17487/rfc1950.
33. Roelofs, G. Zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library / G. Roelofs, J. Gailly, M. Adler. URL: http://www.zlib.net/ (дата обращения: 20.09.2015).
34. Deutsch, P. DEFLATE Compressed Data Format Specification version 1.3 / P. Deutsch. — United States: RFC Editor. — 1996. DOI: 10.17487/rfc1951.
35. TPC Benchmark H — Standard Specification, Version 2.17.1. — Transaction Processing Performance Council (http://www.tpc.org). — 2014. — 136 p. URL: http://www.tpc.org/
tpc_documents_current_versions/pdf/tpch2.17.1.pdf (дата обращения: 29.10.2015).
36. Gray, J. Quickly generating billion-record synthetic databases / J. Gray, P. Sundaresan, S. Englert, K. Baclawski, P.J. Weinberger // Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, May 24–27, 1994, Minneapolis, Minnesota. — ACM Press, 1994. — P. 243–252. DOI: 10.1145/191843.191886.
37. Ungerer, T. A survey of processors with explicit multithreading / T. Ungerer, B. Robič, J. Šilc // ACM Computing Surveys. — 2003. — Vol. 35, No. 1. — P. 29–63. DOI: 10.1145/641865.641867.

Иванова Елена Владимировна, программист отдела поддержки и обучения пользователей Лаборатории суперкомпьютерного моделирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), Elena.Ivanova@susu.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 10 сентября 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 4, pp. 5–31

DOI: 10.14529/cmse150401

COLUMNAR DATABASE COPROCESSOR FOR COMPUTING CLUSTER SYSTEM

E.V. Ivanova, South Ural State University (Chelyabinsk, Russian Federation) Elena.Ivanova@susu.ru,

L.B. Sokolinsky, South Ural State University (Chelyabinsk, Russian Federation) Leonid.Sokolinsky@susu.ru

The paper is devoted to the design and implementation issues of columnar coprocessor for RDBMS. Columnar coprocessor (CCOP) is developed on the base of columnar data storage model. It is designed for large computing cluster systems. CCOP can utilize CPUs as well as manycore coprocessors MIC. CCOP maintains the columnar indices with surrogate keys stored in distributed main memory. The partitioning is performed on the base of domain-interval model. For the data warehouse workload, CCOP demonstrates performance much higher than row-stores do.

Keywords: columnar coprocessor, CCOP, distributed columnar indices, domain-interval fragmentation, computing cluster systems, manycore coprocessors, MIC architecture.

References

1. Chernyshev G.A Organizaciya fizicheskogo urovnya kolonochnyh SUBD [Physical Layer Organization of Columnar DBMS] // Trudy SPIIRAN [Proceedings of the SPIIRAS]. 2013. Vol. 7. No. 30. P. 204–222.
2. Abadi D.J., Boncz P.A., Harizopoulos S., Idreos S., Madden S. The Design and Implementation of Modern Column-Oriented Database Systems // Foundations and Trends in Databases. 2013. Vol. 5, No. 3. P. 197–280. DOI: 10.1561/1900000024.
3. Idreos S., Groffen F., Nes N., Manegold S., Mullender S., Kersten M.L. MonetDB: Two Decades of Research in Column-oriented Database Architectures // IEEE Data Engineering Bulletin. 2012. Vol. 35, No. 1. P. 40–45.
4. Boncz P.A., Zukowski M., Nes N. MonetDB/X100: Hyper-pipelining query execution // Proceedings of the Second Biennial Conference on Innovative Data Systems Research (CIDR), January 4–7, Asilomar, CA, USA. 2005. P. 225–237.
5. Stonebraker M., Abadi D.J., Batkin A., Chen X., Cherniack M., Ferreira M., Lau E., Lin A., Madden S.R., O'Neil E.J., O'Neil P.E., Rasin A., Tran N., Zdonik S.B. C-Store: A Column-Oriented DBMS // Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05), August 30 – September 2, 2005, Trondheim, Norway. ACM, 2005. P. 553–564.
6. MacNicol R., French B. Sybase IQ multiplex — designed for analytics // Proceedings of the Thirtieth International Conference on Very Large Data Bases, August 31 – September 3, 2004, Toronto, Canada. Morgan Kaufmann, 2004. P. 1227–1230. DOI: 10.1016/b978-012088469-8/50111-x.

7. Zukowski M., Boncz P.A. Vectorwise: Beyond column stores // IEEE Data Engineering Bulletin. 2012. Vol. 35, No. 1. P. 21–27.
8. Lamb A., Fuller M., Varadarajan R., Tran N., Vandier B., Doshi L., Bear C. The Vertica analytic database: C-store 7 years later // Proceedings of the VLDB Endowment. 2012. Vol. 5, No. 12. P. 1790–1801. DOI: 10.14778/2367502.2367518.
9. Barber R., Bendel P., Czech M., Draese O., Ho F., Hrle N., Idreos S., Kim M.-S., Koeth O., Lee J.-G., Li T.T., Lohman G. M., Morfonios K., Müller R., Murthy K., Pandis I., Qiao L., Raman V., Sidle R., Stolze K., Szabo S. Business Analytics in (a) Blink // IEEE Data Engineering Bulletin. 2012. Vol. 35, No. 1. P. 9–14.
10. Larson P.-A., Clinciu C., Fraser C., Hanson E. N., Mokhtar M., Nowakiewicz M., Papadimos V., Price S. L., Rangarajan S., Rusanu R., Saubhasik M. Enhancements to SQL server column stores // Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data (SIGMOD'13), June 22–27, 2013, New York, NY, USA. ACM, 2013. P. 1159–1168. DOI: 10.1145/2463676.2463708.
11. Larson P.-A., Clinciu C., Hanson E. N., Oks A., Price S. L., Rangarajan S., Surna A., Zhou Q. SQL server column store indexes // Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (SIGMOD'11), June 12–16, 2011, Athens, Greece. ACM, 2011. P. 1177–1184. DOI: 10.1145/1989323.1989448.
12. Larson P.-A., Hanson E. N., Price S. L. Columnar Storage in SQL Server 2012 // IEEE Data Engineering Bulletin. 2012. Vol. 35, No. 1. P. 15–20.
13. Färber F., May N., Lehner W., Große P., Müller I., Rauhe H., Dees J. The SAP HANA Database – An Architecture Overview // IEEE Data Engineering Bulletin. 2012. Vol. 35, No. 1. P. 28–33.
14. Weiss R. A Technical Overview of the Oracle Exadata Database Machine and Exadata Storage Server. Oracle Corporation White Paper, 2012. 35 p. URL: http://www.oracle.com/technetwork/database/exadata/exadata-technical-whitepaper-134575.pdf (accessed: 29.10.2015).
15. A Drill-Down into EXASolution. Technical Whitepaper. EXASOL AG, 2014. 15 p. URL: http://info.exasol.com/whitepaper-exasolution-2-en.html (accessed: 22.10.2015).
16. A Peek under the Hood. Technical Whitepaper. EXASOL AG, 2014. 16 p. URL: http://www.breos.com/sites/default/files/pdf/downloads/exasol_whitepaper.pdf (accessed: 22.10.2015).
17. EXASolution. Business Whitepaper. EXASOL AG, 2015. 11 p. URL: http://info.exasol.com/business-whitepaper-exasolution-en.html (accessed: 27.10.2015).
18. Actian SQL Analytics in Hadoop. A Technical Overview. Actian Corporation, 2015. 16 p. URL: http://bigdata.actian.com/SQLAnalyticsinHadoop (accessed: 27.10.2015).
19. Ślęzak D., Kowalski M. Towards approximate SQL: infobright's approach // Proceedings of the 7th international conference on Rough sets and current trends in computing (RSCTC'10). Springer-Verlag, 2010. P. 630–639. DOI: 10.1007/978-3-642-13529-3_67.
20. SAND CDBMS: A Technological Overview. White Paper. SAND Technology, 2010. 16 p. URL: http://www.sand.com/downloads/side2239eadd/wp_sand_cdbms_technological_overview_en.pdf (accessed: 29.10.2015).
21. Abadi D.J., Madden S.R, Hachem N. Column-Stores vs. Row-Stores: How Different Are They Really? // Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 9–12, 2008, Vancouver, BC, Canada. ACM, 2008. P. 967–980. DOI: 10.1145/1376616.1376712.

22. Ivanova E., Sokolinsky L. Decomposition of Natural Join Based on Domain-Interval Fragmented Column Indices // Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO, May 25–29, 2015, Opatija, Croatia. IEEE, 2015. P. 223–226. DOI: 10.1109/mipro.2015.7160266.
23. Ivanova E.V., Sokolinsky L.B. Dekompoziciya operacii gruppirovki na baze raspredelennyh kolonochnyh indeksov [Decomposition of Grouping Operation Based on Fragmented Column Indices] // Nauka YUUrGU [Science of SUSU]. Chelyabinsk: SUSU publishing center, 2015. P. 15–23.
24. Ivanova E.V., Sokolinsky L.B. Dekompoziciya operacij peresecheniya i soedineniya na osnove domenno-interval'noj fragmentacii kolonochnyh indeksov [Decomposition of Intersection and Join Operations Based on Domain-Interval Fragmented Column Indices] // Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya «Vychislitel'naya matematika i informatika» [Bulletin of South Ural State University. Series: Computational Mathematics and Software Engineering]. 2015. Vol. 4, No. 1. P. 44–56. DOI: 10.14529/cmse150104.
25. Ivanova E.V. Ispol'zovanie raspredelennyh kolonochnyh hesh-indeksov dlya obrabotki zaprosov k sverhbol'shim bazam dannyh [Using Distributed Column Hash Indices for Query Execution for Very Large Databases] // Nauchnyj servis v seti Internet: mnogoobrazie superkomp'yuternyh mirov: Trudy Mezhdunarodnoj superkomp'yuternoj konferencii [Scientific service in Internet: The variety of supercomputing worlds: Proceedings of the International Supercomputer Conference], September 22–27, 2014, Novorossiysk, Russia. Moscow: MSU publishing center, 2014. P. 102–104.
26. Huffman D. A method for the construction of minimum-redundancy codes // Proceedings of the I.R.E. 1952.Vol. 40, No. 9. P. 1098–1101. DOI: 10.1109/jrproc.1952.273898.
27. Ziv J., Lempel A., A universal algorithm for sequential data compression // IEEE Transactions on Information Theory. 1977. Vol. 23, No. 3. P. 337–343. DOI: 10.1109/tit.1977.1055714.
28. Abadi D. J., Madden S. R., Ferreira M. Integrating compression and execution in column-oriented database systems // Proceedings of the 2006 ACM SIGMOD international conference on Management of data, June 26–29, 2006, Chicago, Illinois. ACM, 2006. P. 671–682. DOI: 10.1145/1142473.1142548.
29. Bassiouni M. A. Data Compression in Scientific and Statistical Databases // IEEE Transactions on Software Engineering. 1985. Vol. 11, No. 10. P. 1047–1058. DOI: 10.1109/tse.1985.231852.
30. Ruth S.S., Kreutzer P.J. Data Compression for Large Business Files // Datamation. 1972. Vol. 19, No. 9. P. 62–66.
31. Roth M. A., Van Horn S. J. Database compression // ACM SIGMOD Record. 1993. Vol. 22, No. 3. P. 31–39. DOI: 10.1145/163090.163096.
32. Deutsch P., Gailly J.-L. ZLIB Compressed Data Format Specification version 3.3. United States: RFC Editor, 1996. DOI: 10.17487/rfc1950.
33. Roelofs G., Gailly J., Adler M. Zlib: A Massively Spiffy Yet Delicately Unobtrusive Compression Library. URL: http://www.zlib.net/ (accessed: 20.09.2015).
34. Deutsch P. DEFLATE Compressed Data Format Specification version 1.3. United States: RFC Editor, 1996. DOI: 10.17487/rfc1951.

35. TPC Benchmark H — Standard Specification, Version 2.17.1. Transaction Processing Performance Council (http://www.tpc.org), 2014. 136 p. URL: http://www.tpc.org/tpc_documents_current_versions/pdf/tpch2.17.1.pdf (accessed: 29.10.2015).
36. Gray J., Sundaresan P., Englert S., Baclawski K., Weinberger P. J. Quickly generating billion-record synthetic databases // Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, May 24–27, 1994, Minneapolis, Minnesota. ACM Press, 1994. P. 243–252. DOI: 10.1145/191843.191886.
37. Ungerer T., Robič B., Šilc J. A survey of processors with explicit multithreading // ACM Computing Surveys. 2003. Vol. 35, No. 1. P. 29–63. DOI: 10.1145/641865.641867.

Received September 10, 2015.

УДК 004.93'11, 004.932.2 DOI: 10.14529/cmse150402

МЕТОДИКИ СОПОСТАВЛЕНИЯ ОСОБЫХ ТОЧЕК В ЗАДАЧЕ ВИЗУАЛЬНОЙ НАВИГАЦИИ БПЛА

Д.Н. Степанов

Статья посвящена разработке и экспериментальному сравнению методик сопоставления особых точек на изображениях — снимках земной поверхности с камер, установленных на беспилотном летательном аппарате (БПЛА) и искусственном спутнике Земли. Главная особенность задачи состоит в том, что одно из изображений (спутниковый снимок) разбит на фрагменты. Разработанные методики являются частью комплекса алгоритмов, предназначенных для определения положения и ориентации БПЛА с использованием средств, методов и алгоритмов технического зрения. Приведено описание технологии моделирования полета и решения задачи позиционирования. Особые точки на изображениях выделяются с помощью алгоритма SURF. Также исследован подход к сопоставлению, основанный на разбиении множества особых точек на два подмножества в зависимости от знака лапласиана. Предложены способы увеличения производительности сопоставления точек.

Ключевые слова: БПЛА, особые точки, SURF, компьютерное зрение, спутниковые снимки, полный перебор, поисковый индекс, сопоставление изображений.

Введение

В настоящее время сфера применения беспилотных летательных аппаратов (БПЛА) достаточно широка и включает в себя как военные, так и гражданские области. Создание надежных систем навигации и управления БПЛА является актуальным и важным направлением разработок. Широко применяемые радионавигационной системы GPS/ГЛОНАСС имеют свой набор недостатков (отсутствие автономности, потеря и затухание сигнала, естественные и искусственные помехи, вызванные погодой или рельефом местности и т.д.). Инерциальные навигационные микромеханические системы способны работать автономно, но их использование связано с проблемой накопления ошибки в вычислениях положения и ориентации.

Одним из альтернативных способов навигации является использование средств, методов и алгоритмов технического зрения. В 2011–2012 годах в ИПС им. А. К. Айламазяна РАН проводилась научно-исследовательская работа, в рамках которой был разработан алгоритм решения задачи позиционирования БПЛА по видеоряду для полетов над плоской местностью, основанный на сопоставлении так называемых особых точек, найденных на кадре и на спутниковом снимке того участка местности, над которой БПЛА совершает свои полеты. После сопоставления, вычисляются параметры проективного преобразования между двумя изображениями, что позволяет вычислить положение и ориентацию камеры на БПЛА в глобальной системе координат. Проведение экспериментов с БПЛА возможно далеко не всегда из-за ограничений российского законодательства (намеки на некоторое улучшение ситуации появились совсем недавно [1]). Поэтому разработанные алгоритмы тестировались на виртуальных видеорядах — результатах моделирования полета БПЛА с помощью методов компьютерной графики.

Задача выделения и сопоставления особенностей на изображениях достаточно хорошо рассмотрена в литературе. Один из наиболее важных критериев, по которому определяется эффективность алгоритмов выделения и сопоставления особых точек —

устойчивость к изменениям яркости, аффинным преобразованиям (поворот, масштабирование), проективным преобразованиям. Для сопоставления особенностей с разных изображений каждой особой точке приписывается некоторый дескриптор (описатель). Простейший дескриптор – сама окрестность точки, но такой подход будет работать, если окрестность точки предварительно нормализована по масштабу, ориентации и яркости (или же для обеспечения инвариантности требуется много шаблонов в разных ориентациях). Для сравнения окрестностей двух точек используются подходы, основанные на корреляции [2].

Одними из простейших алгоритмов выделения особых точек являются алгоритмы выделения углов на изображения: примером здесь является детектор углов Харриса [3]. Но детекторы углов не инвариантны к масштабированию: при разных масштабах одного и того же изображения на них могут быть найдены разные угловые точки. Более устойчивы алгоритмы, в которых используется так называемое scale-space представление изображения: размытие изображения при помощи свертки с функцией Гаусса с разными значениями среднеквадратического отклонения [4]: SIFT [5], SURF [6], GLOH [7] и т.д. Во всех этих алгоритмах для каждой особой точки вычисляется ее дескриптор, который в определенной мере инвариантен к изменениям яркости, аффинным и перспективным преобразованиям.

Существует набор алгоритмов, связанных с вычислением так называемого оптического потока, классическим подходом здесь является алгоритм Лукаса—Канаде и его расширенная пирамидальная версия, в которой используется представление изображения в нескольких масштабах и которая характеризуется большей устойчивостью к изменению масштаба [8]. Но алгоритмы оптического потока изначально были разработаны для сопоставления точек с двух изображений, которые незначительно различаются между собой (чего нельзя сказать о снимках с ИСЗ и БПЛА).

Для сопоставления дескрипторов особых точек с двух изображений используются следующие методы:

- полный перебор: для каждого дескриптора из первого набора вычисляется расстояния до всех дескрипторов со второго изображения и берется лучшая;
- ускоренные приближенные метод с использованием иерархических структур данных. В этих методах строится так называемый поисковый индекс для поиска ближайших соседей в многомерном пространстве. Примером такой структуры данных является kd-дерево [9].

Если же рассмотреть задачу выделения и сопоставления особых точек с изображений применительно к фотоснимкам земной поверхности (например, с искусственного спутника Земли — ИСЗ), то мы приходим к понимаю следующей проблемы: эталонные снимки земной поверхности могут иметь очень большие размеры, и непосредственное сопоставление особых точек со всего эталонного снимка может быть крайне ресурсозатратной операцией (особенно с учетом вычислительных возможностей бортовых компьютеров). Если строить для всего набора эталонных данных поисковый индекс, то его представление в оперативной памяти так же будет весьма значительным по размеру. Для решения проблемы в рамках НИР было предложено разбиение спутникового снимка на независимые фрагменты. Также был разработан алгоритм сужения области поиска [10], который основан на определении координат и размеров той области спутникового снимка, которая запечатлена на текущем кадре с борта БПЛА.

Цель настоящего исследования — развитие и совершенствование технологии позиционирования БПЛА с помощью методов и алгоритмов технического зрения. Задачи исследования — разработать методики эффективного сопоставления особых точек со снимков с ИСЗ и БПЛА, а также исследовать эффективность их программных реализаций. В разделе 1 дано общее описание технологии моделирования полета БПЛА и решения задачи визуальной навигации. В разделе 2 описаны предложенные методики сопоставления особых точек. В разделе 3 предложены способы сравнения методик на предмет эффективности. В разделе 4 приведены результаты экспериментальных исследований. В заключении приведены результаты сравнения методик, а также возможные способы увеличения их производительности.

1. Описание технологии моделирования полета и решения задачи позиционирования

Опишем технологию моделирования полета виртуального БПЛА и его позиционирования с помощью методов и алгоритмов технического зрения.

Этап подготовки данных:

1. Для исходного спутникового снимка строится так называемая пирамида изображений, при этом каждый слой пирамиды разбивается на фрагменты (тайлы, англ. tiles).
2. На каждом тайле выполняется поиск особых точек с помощью алгоритма SURF, а также вычисляются их дескрипторы (каждый дескриптор — вектор из 64 вещественных чисел).
3. Среди особых точек, найденных на тайле, выбирается несколько точек, для каждой из которых вычисляется еще один дескриптор, состоящий из откликов различных вейвлетов Габора [11], а также вычисляются трехмерные координаты этой точки. Эти несколько точек назовем опорными.
4. Все вычисленные данные сохраняются на жесткий диск для дальнейшего использования в рамках решения задачи визуальной навигации.

Моделирование полета посредством порождения виртуального видеоряда:

1. Пользователь задает желаемую траекторию полета БПЛА, отмечая на уменьшенной копии спутникового снимка ряд опорных точек, которые затем соединяются прямыми отрезками.
2. На каждом отрезке выбирается ряд промежуточных точек, которые будут соответствовать тем моментам времени, когда камера на БПЛА сняла очередной кадр видеоряда.
3. Для каждой промежуточной точки вычисляется ориентация камеры на БПЛА (при условии, что в случае горизонтального полета камера направлена перпендикулярно поверхности Земли) и генерируется кадр видеоряда, с использованием спутниковых снимков местности и (возможно) цифровых карт высот.

Собственно, этап решения задачи навигации состоит в выполнении следующих действий для каждого кадра видеоряда:

1. С помощью алгоритма SURF на кадре выделяются особые точки, вычисляются их дескрипторы.
2. Исходя из положения БПЛА в предыдущий момент времени, с помощью алгоритма сужения области поиска [10, 12] выбираются подходящий слой пирамиды спут-

никовых снимков, а также набор тайлов из этого слоя, особые точки с которых будут использоваться для сопоставления с особыми точками на кадре с БПЛА.

3. Выполняется сопоставление особых точек со снимка с БПЛА и тайлов спутникового снимка. На выходе имеем множество пар координат особых точек, при этом часть пар могут быть ложными (в силу того, что какие-то особые точки на снимках с БПЛА или ИСЗ могут иметь схожие дескрипторы).
4. С помощью 5-точечного алгоритма [13], объединенного с методикой RANSAC [14] вычисляется фундаментальная матрица F, связывающая координаты особых точек на снимках с БПЛА и ИСЗ. Методика RANSAC позволяет устранить часть ложных соответствий. Производится декомпозиция матрицы F на матрицу поворота R (определяет взаимную ориентацию камер на БПЛА и ИСЗ) и вектор T (определяет взаимное положение камер, но с точностью до некоторого неизвестного положительного коэффициента).
5. С помощью матрицы R и вектора T производится геометрическая коррекция снимка с БПЛА для проведения процедуры поиска опорных точек со спутникового снимка на снимке с БПЛА с применением фильтров (вейвлетов) Габора. Детальное описание процедуры будет приведено в следующих публикациях. Результатом поиска является набор пар вида «двумерные координаты опорной точки на снимке с БПЛА — трехмерные координаты этой точки в глобальной системе координат». Это позволяет свести задачу позиционирования к задаче внешней калибровки камеры на БПЛА (вычисления ее положения и ориентации в глобальной системе координат) [15]. Поскольку на этапе генерации видеоряда мы изначально знали положение и ориентацию камеры на БПЛА для каждого кадра, эти данные можно использовать для проверки точности навигации путем вычисления расстояние между истинным положением БПЛА и результатом работы алгоритма позиционирования.

2. Методики сопоставления особых точек

В реализации алгоритма сопоставления точек, который использовался в рамках выполненного проекта, было разработано два подхода к сопоставлению, оба подхода основаны на том, что исходный спутниковый снимок разбивается на тайлы и представлен в нескольких масштабах. Позже было разработано еще четыре методики сопоставления. Опишем все шесть разработанных методик.

1. Для каждого тайла B_i (пусть их общее количество равно N) выполняется полный перебор всех пар особых точек, найденных на видеокадре и на тайле B_i Для каждой точки $A[j]$ выбирается такая точка $B_i[number]$, что $number = \arg\min_k \left(\text{eucl\_dist}\left(dA[j], dB_i[k]\right)\right)$. Здесь dA — массив дескрипторов особых точек с видеокадра, dB_i — аналогичный массив для тайла B_i, eucl_dist — функция вычисления евклидова расстояния между двумя дескрипторами. Если найденная точка лежит вне рассматриваемой области поиска, которая представляет собой круг [1, 4], то она отбрасывается. Недостатком подхода является то, что если среди тайлов $B_1 \ldots B_N$ есть такой тайл B_j, что между ним и видеокадром заведомо нет общих особых точек (на них запечатлены непекрывающиеся фрагмен-

ты поверхности Земли), то сопоставление может быть проведено неверно. Псевдокод алгоритма представлен на рис. 1.

```
для каждого тайла Bi
{
   для каждой особой точки A[j]
   {
      min_dist = +infinity
      index = 0
      для каждой особой точки Bi[k] ← Bi
      {
         d = eucl_dist(dA[j], dBi[k])
         если d < min_dist то
         {
            min_dist = d
            index = k
         }
      }
      если point_in_area(Bi[index]) то
         add_to_result(A[j], Bi[index])
   }
}
```

Рис. 1. Псевдокод методики 1

Функция point_in_area проверяет, попадает ли точка $B_i[index]$ в область поиска, функция add_to_result добавляет точки $A[j]$ и $B_i[index]$ в массив-результат работы алгоритма.

2. Каждая особая точка из видеокадра сопоставляется со всеми особыми точками со всех тайлов. Такой подход свободен от недостатка, когда в ходе сопоставления используется тайл, с которым у видеокадра с БПЛА заведомо нет пересечения по общим особым точкам. Псевдокод представлен на рис. 2.

```
для каждой особой точки A[j]
{
   min_dist = +infinity
   index1 = 0
   index2 = 0
   для каждого тайла Bi
   {
      для каждой особой точки Bi[k] ← Bi
      {
         если point_in_area(Bi[k]) то
         {
            d = eucl_dist(dA[j], dBi[k])
            если d < min_dist то
            {
               min_dist = d
               index1 = i
               index2 = k
            }
         }
      }
   }
   add_to_result(A[j], Bindex1[index2])
}
```

Рис. 2. Псевдокод методики 2

3. Для каждого тайла B_i выполняется следующее: для всех точек из B_i выполняется полный перебор всех точек из видеокадра. При этом в ходе сопоставления пропускаются те точки с тайлов, которые лежат вне области поиска. Псевдокод представлен на рис. 3.

```
для каждого тайла Bi
{
   для каждой особой точки Bi[k] ← Bi
   {
      если point_in_area(Bi[k]) то
      {
         min_dist = +infinity
         index = 0
         для каждой особой точки A[j]
         {
            d = eucl_dist(dA[j], dBi[k])
            если d < min_dist то
            {
               min_dist = d
               index = j
            }
         }
         add_to_result(A[index], Bi[k])
      }
}
```

Рис. 3. Псевдокод методики 3

4. Для каждого тайла B_i выполняется следующее: для множества дескрипторов особых точек, найденных на нем, строится поисковый индекс [16]. Далее, для дескриптора каждой точки из видеокадра находится ближайший к ней дескриптор из B_i с помощью поискового индекса. После проведения сопоставления с поисковым индексом, отбрасываются те точки, которые лежат вне рассматриваемой области поиска. Псевдокод представлен на рис. 4.

```
для каждого тайла Bi
{
   S = make_FLANN_index(dBi)
   для каждой особой точки A[j]
   {
      index = search(S, dA[j])
      если point_in_area(Bi[index]) то
         add_to_result(A[j], Bi[index])
   }
}
```

Рис. 4. Псевдокод методики 4

Функция make_FLANN_index строит поисковый индекс, функция search находит для дескриптора $dA[j]$ наиболее близкий к нему дескриптор с тайла B_i с помощью поискового индекса. Все поисковые индексы можно построить и сохранить заранее, в процессе выделения особых точек на тайлах пирамиды спутниковых снимков.

5. Эта методика также основана на использовании поискового индекса, но здесь он строится для множества дескрипторов, которое является объединением дескрип-

торов всех особых точек со всех тайлов $B_1 \ldots B_N$. После проведения сопоставления с поисковым индексом, отбрасываются те точки, которые лежат вне рассматриваемой области поиска в виде круга. Псевдокод представлен на рис. 5.

```
S = make_FLANN_index(dB1, dB2, … dBN)

для каждой особой точки A[j]
{
   (index1, index2) = search_2(S, dA[j])
     если point_in_area(B_index1[index2]) то
       add_to_result(A[j], B_index1[index2])
   }
}
```

Рис. 5. Псевдокод методики 5

Конечно, здесь дополнительные ресурсы тратятся на создание индекса, но данная методика имеет то преимущество, что если для позиционирования БПЛА по текущему кадру требуется использовать те же самые тайлы пирамиды спутниковых снимков, что и для предыдущего кадра, то можно применить тот же самый поисковый индекс для предыдущего кадра.

6. Поисковый индекс строится для множества дескрипторов особых точек, найденных на видеокадре. Затем, для каждого тайла B_i выполняется следующее: для всех точек из B_i с помощью индекса находится ближайшая точка из видеокадра (ближайшая — в смысле расстояния между их дескрипторами). Затем отбрасываются те точки с тайла B_i, которые лежат вне области поиска. Псевдокод представлен на рис. 6.

```
S = make_FLANN_index(dA)

для каждого тайла Bi
{
   для каждой особой точки Bi[k] ← Bi
   {
      index = search(S, dBi[k])
      если point_in_area(Bi[k]) то
        add_to_result(A[index], Bi[k])

   }
}
```

Рис. 6. Псевдокод методики 6

Все шесть методик комбинировались с подходом, взятым из статьи [5] и позволяющим частично устранить пары неправильно найденных точек. Пусть есть два набора дескрипторов особых точек. Для каждого дескриптора p_i из первого набора находятся две наиболее подходящие ей точки q_{j1} и q_{j2} из второго набора (исходя из критерия минимизации евклидова расстояния между дескрипторами). Пусть $d_1 = \text{eucl\_dist}\left(p_i, q_{j1}\right)$, $d_2 = \text{eucl\_dist}\left(p_i, q_{j2}\right)$, $d_2 < d_2$. Если $d_2 / d_2 < thresh$ (где $thresh$ — некоторое пороговое значение от *0* до *1*), то точка p_i сопоставляется точке q_{j1}, иначе считается, что для точки p_i найти пару не удалось.

3. Способы сравнения разработанных методик

Сравнение методик сопоставления особых точек только по критерию быстродействия является недостаточным, поскольку важнейшим показателем в конечном счете является точность позиционирования БПЛА (погрешность в вычислении положения и ориентации). Точность зависит от качества исходных данных для алгоритма решения задачи внешней калибровки камеры (использовалась реализация из библиотеки OpenCV [17]), которая, в свою очередь, зависит от результатов работы алгоритма поиска опорных точек на видеокадре с помощью вейвлетов Габора (будет описан в следующих публикациях). А результаты его работы зависят от того, насколько точно с помощью 5-точечного алгоритма и методики RANSAC вычислена фундаментальная матрица, связывающая изображения с БПЛА и ИСЗ.

Используемая реализация алгоритма вычисления фундаментальной матрицы позволяет не только найти матрицу F, но и пометить пары соответствующих точек, которые не удовлетворяют фундаментальной матрице (ложные срабатывания алгоритма сопоставления точек). Эксперименты показали, что если после вычисления матрицы F остается слишком мало пар точек (как правило, менее 20-25), то матрица F в подавляющем большинстве случаев оказывается неправильной.

Таким образом, были сформулированы следующие критерии сравнения методик сопоставления особых точек:

1. Время сопоставления особых точек.
2. Количество найденных пар точек.
3. Время вычисления фундаментальной матрицы.
4. Количество пар особых точек, которые после вычисления матрицы F не были помечены как ложные (так называемые «инлайеры», англ. inliers).
5. Итоговая погрешность в определении координат БПЛА.

В алгоритме SURF, как и во многих других алгоритмах обработки и анализа изображений, исходное изображение рассматривается как дискретная двумерная функция яркости. Одним из описателей особой точки является знак лапласиана $sign(\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2})$ (положительный или отрицательный). Как утверждают авторы алгоритма, если две особые точки с двух изображений имеют разный знак лапласиана, то они гарантированно не могут являться парой соответствующих точек. Таким образом, множество особых точек (и их дескрипторов) можно разбить на две группы, в зависимости от знака лапласиана, и сопоставлять между собой наборы точек, имеющих один и тот же знак. Было решено реализовать и исследовать этот подход на примере шести предложенных методик сопоставления точек.

4. Экспериментальное сравнение разработанных методик

Для детектирования особых точек и вычисления дескрипторов с помощью алгоритма SURF использовалась библиотека OpenCV, применялась однопоточная версия алгоритма, который выполнялся на одном из ядер процессора Intel Xeon E5410. Разрешение сгенерированных видеокадров с виртуального БПЛА составляло 1024*768 пикселей. Реализация алгоритма SURF из библиотеки OpenCV находила на снимках с виртуального БПЛА примерно от 2700 до 3600 особых точек (время счета — 1,5–1,7 с). Количество

используемых тайлов одного из слоев пирамиды спутниковых снимков составляло $6 \cdot 6 = 36$ тайлов, каждый размером $256x256$ пикселей. Количество особых точек на каждом тайле варьировалось примерно от 50 до 700, зависело от текстуры земной поверхности и от количества деталей на изображениях. Пространственное разрешение спутникового снимка — 1 метр на пиксель.

Рис. 7 иллюстрирует задачу сопоставления особых точек: сверху представлен кадр видеоряда с БПЛА, снизу — часть тайлов одного из слоев спутникового снимка, для наглядности тайлы разделены белыми вертикальными и горизонтальными линиями. Пары соответствующих особых точек на снимке с БПЛА и ИСЗ соединены отрезками.

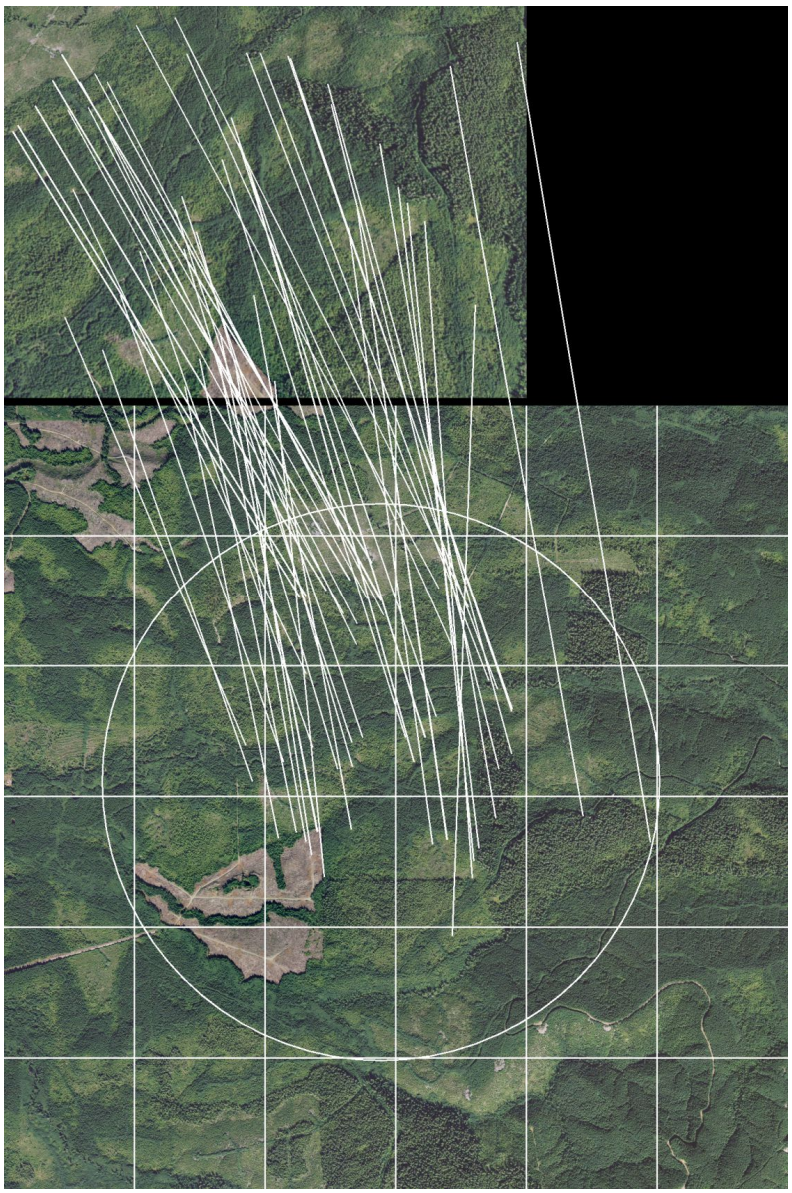


Рис. 7. Сопоставление особых точек с видеокадра и тайлов спутникового снимка

Для тестирования использовались три траектории полета виртуального БПЛА с генерацией видеоряда с учетом рельефа Земли (карты высот), а также три аналогичных траектории, но с генерацией видеоряда при полете над плоской местностью:

1. Горизонтальный полет в северном направлении на постоянной высоте над плоской местностью.

2. Горизонтальный полет в северном направлении на постоянной высоте над местностью с рельефом.
3. Горизонтальный полет в северо-восточном направлении над плоской местностью. Здесь можно тестировать инвариантность алгоритма SURF к поворотам изображений.
4. Горизонтальный полет в северо-восточном направлении над местностью с рельефом.
5. Полет в северо-восточном направлении с набором высоты над плоской местностью. Здесь можно тестировать инвариантность алгоритма SURF к поворотам, масштабированию и проективным преобразованиям.
6. Полет в северо-восточном направлении с набором высоты над местностью с рельефом. На рис. 7 как раз и приведен результат сопоставления одного из кадров такого видеоряда с тайлами спутникового снимка.

Сразу отметим, что методика под номером 4 оказалась неспособной корректно сопоставлять особые точки: время сопоставления было слишком велико (3 секунды и более для одного кадра видеоряда), она находила слишком много ложных соответствий, что негативно сказывалось на времени вычисления фундаментальной матрицы (несколько секунд и даже несколько десятков секунд для одного кадра). Результаты вычисления матрицы F во всех случаях оказывались весьма далеки от эталонных значений, что делало невозможным решение задачи визуальной навигации.

Во всех остальных случаях задача позиционирования решалась довольно успешно, погрешность в вычислении положения БПЛА составляла от 1,5 до 3,3 метра (за исключением ряда случаев).

В табл. 1 приведено сравнение пяти методик, для каждой из шести тестовых траекторий полета. По времени сопоставления наиболее эффективны методики 5 и 6 (основанные на использовании поисковых индексов), затем идут методики 2 и 3, наименее эффективна методика 1. По времени вычисления матрицы F наиболее эффективны методики 2 и 3, затем идут методики 1 и 5, наименее эффективна методика 6. Но в целом, для всех подходов характерен недостаток, что в случае наличия значительных проективных искажений (тест №6) между эталонными данными (спутниковый снимок) и тестовыми (видеокадр с БПЛА) удается найти недостаточно пар-«инлайеров», что не позволяет решить задачу позиционирования.

Перейдем к сравнению предложенных методик в случае, когда особые точки с видеокадра разбивались на две группы, в зависимости от знака лапласиана (табл. 2). Процессорное время, необходимое на разбиение, оказалось невелико (не более 1 миллисекунды), поэтому это не повлияло на быстродействие всех шести методик. По времени сопоставления наиболее эффективны методики 5 и 6 (основанные на использовании поисковых индексов), затем идут методики 2 и 3, наименее эффективна методика 1. По времени вычисления матрицы F наиболее эффективны методики 2 и 3, затем идут методики 1 и 5, наименее эффективна методика 6. Как видим, разбиение множества особых точек на два подмножества по знаку лапласиана позволило повысить количество найденных пар примерно в 1,5–2 раза, увеличилось и количество найденных пар-«инлайеров», что повысило точность вычисления фундаментальной матрицы (но даже в этом случае методика 6 дала сбой).

Таблица 1

Сравнение методик сопоставления особых точек

Номер методики	Время сопоставления, с	Кол-во найденных пар точек	Время вычисления матрицы F, с	Кол-во пар точек-инлайеров	Примечание
1	1,0–1,25	632–654 403–468 200–225 171–191 132–142 120–140	0,12–0,27 0,28–0,55 0,21–0,51 0,70–0,95 0,05–0,33 0,12–0,28	400–430 200–240 60–83 49–61 25–46 14–29	В тесте №6 инлайеров оказалось мало, ошибка в позиционировании составила 19 м.
2	0,7–1	486–510 247–292 80–102 54–69 38–47 24–33	0,06–0,21 0,08–0,17 0,06–0,10 0,03–0,07 0,02–0,04 0,02–0,05	345–384 127–164 45–54 30–35 20–26 15–21	
3	0,50–0,75	515–538 273–309 100–134 70–80 34–46 25–38	0,08–0,15 0,09–0,19 0,06–0,18 0,10–0,14 0,04–0,07 0,03–0,05	326–403 162–188 42–66 30–35 18–26 12–22	В тесте №6 инлайеров оказалось мало, ошибка в позиционировании составила 8 м.
5	0,15–0,18 (без учета построения индекса поиска — 0,09–0,11)	640–659 420–484 238–240 185–199 129–160 126–148	0,21–0,31 0,26–0,46 0,41–0,49 0,17–1,10 0,02–0,28 0,13–0,31	393–476 195–242 19–102 48–68 33–50 14–34	В тестах №3 и №6 инлайеров оказалось мало, вычислить положение БПЛА не удалось.
6	0,14–0,18	714–738 503–532 279–321 241–276 150–182 159	0,29–0,36 0,28–0,48 0,62–1,59 0,20–1,58 0,14-1,75 0,04	387–425 235–265 96–109 53–81 40–56 12	В тесте №6 инлайеров оказалось мало, вычислить положение БПЛА не удалось.

Таблица 2

Сравнение методик сопоставления особых точек с учетом знака лапласиана

Номер методики	Время сопоставления, с	Кол-во найденных пар точек	Время вычисления матрицы F, с	Кол-во пар точек-инлайеров	Примечание
1	1,0–1,50	1287–1313 844–912 386–419 312–341 214–227 188–203	0,23–0,39 0,39–0,82 0,24–0,92 0,70–1,00 0,25–0,65 0,30–1,10	737–942 391–513 141–174 114–130 72–88 66–78	
2	0,75–1,00	1050–1083 580–638 190–211 133–159 83–105 64–87	0,15–0,39 0,22–0,28 0,13–0,19 0,08–0,14 0,06–0,10 0,03–0,14	732–816 308–372 93–110 65–90 39–53 29–43	
3	0,50–0,75	1103–1132 600–670 218–254 157–175 78–96 54–77	0,11–0,50 0,24–0,50 0,10–0,33 0,11–0,22 0,08–0,15 0,06–0,10	637–894 304–400 99–127 77–85 36–46 29–36	
5	0,2–0,23 (без учета построения индекса поиска — 0,09–0,11)	1316–1352 900–976 423–489 356–357 248–264 227–243	0,28–0,51 0,37–0,62 0,57–1,06 0,15–1,33 0,40–1,09 0,05–1,29	785–1006 424–527 155–211 126–143 82–102 71–83	
6	0,26–0,31	1523–1566 1099–1149 616–674 505–547 349–366 296–338	0,40–0,81 0,64–1,03 0,80–2,63 2,40–3,20 0,40–5,68 1,86–3,88	828–973 515–562 186–229 163–175 73–109 22–93	В тесте №6 инлайеров оказалось мало, вычислить положение БПЛА не удалось.

Заключение

Экспериментальное сравнение разработанных методик сопоставления особых точек с двух изображений, одно из которых разделено на фрагменты, показало, что эффективность и производительность их реализаций зависит от исходных данных: количества особых точек, вида проективного преобразования, которое связывает два изображения (параллельный перенос, поворот, масштабирование), а также наблюдаемой сцены (является ли она плоской или имеет перепады высот). Также оправдал себя подход, основан-

ный на разбиении множества особых точек, найденных алгоритмом SURF, на две группы (в зависимости от знака лапласиана).

Методики под номерами 1, 2, 3 и 5 показали свою эффективность, хотя на данном этапе производительность реализаций методик 1 и 5 не позволяет использовать их в режиме времени, близкого к реальному. В дальнейших экспериментах предполагается опробовать следующие способы увеличения производительности всех методик:

1. Использовать различных значений порогового значения *thresh* (в текущих экспериментах использовалось значение 0,65).
2. Провести эксперименты с параметрами алгоритма SURF для уменьшения времени поиска особых точек.
3. Искусственно уменьшить количества точек, которые подаются на вход алгоритму вычисления фундаментальной матрицы. Слишком большое количество избыточно, оно увеличивает время счета, точность позиционирования при этом почти не меняется. Возможно, стоит из всех пар соответствующих точек в количестве P выбрать случайные Q пар, где Q может быть фиксированным (например, 100–150), или же зависеть от числа особых точек на видеокадре с БПЛА или от числа P.
4. Использовать возможности векторизации кода и распараллеливания по вычислительным ядрам. Конкретные способы увеличения быстродействия зависят от используемого вычислителя, который будет использоваться на борту БПЛА.

Литература

1. Кабмин РФ предложил изменить законодательство для использования БПЛА. URL: http://ria.ru/politics/20150217/1048089808.html (дата обращения: 14.07.2015).
2. Correlation based similarity measures — Summary. URL: https://siddhantahuja.wordpress.com/tag/sum-of-absolute-differences-sad (дата обращения: 14.07.2015).
3. Harris, C. A combined corner and edge detector / C. Harris, M. Stephens // Proceedings of the 4th Alvey Vision Conference — 1988 — P. 147—151. DOI: 10.5244/c.2.23.
4. Гаганов, В. Инвариантные алгоритмы сопоставления точечных особенностей на изображениях /В. Гаганов // Компьютерная графика и мультимедиа — 2009. Выпуск №7(1) — URL: http://cgm.computergraphics.ru/issues/issue17/invariant_features (дата обращения: 06.07.2015)
5. Lowe, D.G. Distinctive image features from scale-invariant keypoints / D.G. Lowe // International Journal of Computer Vision — 2004 — Vol. 60, Issue 2 — P. 91–110. DOI: 10.1023/b:visi.0000029664.99615.94.
6. Bay, H. SURF: Speeded Up Robust Features / H. Bay, A. Ess, T. Tuytelaars, L Van Gool // Computer Vision and Image Understanding (CVIU) — 2008. — Vol. 110, №. 3 — 14 p. DOI: 10.1016/j.cviu.2007.09.014.
7. Mikolajczyk, K. A performance evaluation of local descriptors / K. Mikolajczyk, C. Schmid // IEEE Transactions on Pattern Analysis and Machine Intelligence — 2005 — Vol. 10, №. 27 — P. 1615-1630. DOI: 10.1109/tpami.2005.188.
8. Bouguet, J. Y. Pyramidal implementation of the Lucas Kanade feature tracker / J. Y. Bouguet // Intel Corporation, Microprocessor Research Labs — 2000 — 9 p.

9. Muja, M. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration / M. Muja, D.G. Lowe // International Conference on Computer Vision Theory and Application VISSAPP'09 — 2009 — P. 331-340
10. Степанов, Д.Н., Тищенко, И.П., Поляков, А.В., Ватутин, В.М., Соболев, Д.Б. Подсистема определения положения и ориентации беспилотного летательного аппарата / Д.Н. Степанов, И.П. Тищенко, А.В. Поляков, В.М. Ватутин, Д. Б.Соболев, // Ракетно-космическое приборостроение и информационные технологии. 2012: Труды V Всероссийской научно-технической конференции «Актуальные проблемы ракетно-космического приборостроения и информационных технологий» (Москва, 5-7 июня 2012 г.) / под ред. Ю.М. Урличича, А.А. Романова. — М.: Радиотехника, 2013. - С. 9–27.
11. Kamarainen, J.-K. Invariance Properties of Gabor Filter Based Features — Overview and Applications / J.-K. Kamarainen, V. Kyrki, H. Kälviäinen // IEEE Transactions on Image Processing — 2006. — Vol. 15, No. 5 — 12 p. DOI: 10.1109/tip.2005.864174.
12. Степанов, Д.Н. Методы и алгоритмы определения положения и ориентации беспилотного летательного аппарата с применением бортовых видеокамер / Д.Н. Степанов // Программные продукты и системы (международный журнал) — 2014. — Т. 1, № 1. URL: http://www.swsys.ru/index.php?page=article&id=3776 (дата обращения: 06.07.2015).
13. Stewenius. H. Recent developments on direct relative orientation / H. Stewenius, C. Engels, D. Nister. // ISPRS Journal of Photogrammetry and Remote Sensing — 2006. — Vol. 60, Issue 4 — P. 284–294. DOI: 10.1016/j.isprsjprs.2006.03.005.
14. Fischler, M.A. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography / M.A. Fischler, R.C. Bolles // Comm. Of the ACM 24 — 1981. - P. 381–395. DOI: 10.1145/358669.358692.
15. Кравцов, А. Общая формулировка задачи внешней калибровки камеры / А. Кравцов, В. Вежневец // Компьютерная графика и мультимедиа — 2003. — Выпуск №1(2) URL: http://cgm.computergraphics.ru/content/view/34 (дата обращения: 06.07.2015).
16. FLANN — Fast Library for Approximate Nearest Neighbors. URL: http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN (дата обращения: 06.07.2015).
17. Bradski, G. A. Learning OpenCV / G. Bradski., A. Kaehler // O'Reilly Media — September 2008 — 576 p.

Степанов Дмитрий Николаевич, инженер-исследователь, Исследовательский центр мультипроцессорных систем Института программных систем им. А.К. Айламазяна РАН (Переславль-Залесский, Российская Федерация), mitek1989@mail.com.

Поступила в редакцию 8 июля 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 4, pp. 32–47

DOI: 10.14529/cmse150402

TECHNIQUES OF FEATURE POINTS MATCHING IN THE PROBLEM OF UAV'S VISUAL NAVIGATION

D.N. Stepanov, Aylamazyan Program Systems Institute of the Russian Academy of Sciences (Pereslavl-Zalessky, Russia) mitek1989@mail.com

The paper is devoted to development and experimental comparison of techniques feature points matching on the images — images of the earth's surface with cameras mounted on unmanned aerial vehicles (UAVs) and artificial earth satellite. The main feature of the problem is that one of the images (satellite image) is divided into fragments. The developed techniques are part of a complex of algorithms for determining the position and orientation of the UAV using the methods and algorithms of machine vision. A description of the flight simulation technology and solving positioning tasks are described. Feature points on the image are extracted using an SURF algorithm. Also the approach to matching based on the partition of feature points' set into two subsets depending on the sign of the Laplacian are stidied. Methods of increasing the performance of matching points are offered.

Keywords: UAV, feature points, SURF, computer vision, satellite images, brute-force, search index, image matching.

References

1. Kabmin RF predlozhil izmenit' zakonodatel'stvo dlja ispol'zovanija BPLA [The Cabinet of Ministers of the Russian Federation proposed to change the law for the use of UAVs]. URL: URL: http://ria.ru/politics/20150217/1048089808.html (accessed: 14.07.2015)
2. Correlation based similarity measures — Summary. URL: https://siddhantahuja.wordpress.com/tag/sum-of-absolute-differences-sad (accessed: 14.07.2015).
3. Harris C., Stephens M. A combined corner and edge detector // Proceedings of the 4th Alvey Vision Conference. 1988. P. 147—151. DOI: 10.5244/c.2.23.
4. V. Gaganov. Invariantnye algoritmy sopostavlenija tochechnyh osobennostej na izobrazhenijah [Invariant algorithms of feature points' matching on images] // Computer graphics and multimedia. 2009. Vol. 7, No 1.
URL: http://cgm.computergraphics.ru/issues/issue17/invariant_features (accessed: 14.07.2015).
5. D.G. Lowe, Distinctive image features from scale-invariant keypoints // International Journal of Computer Vision. 2004. Vol. 60, Issue 2. P. 91–110. DOI: 10.1023/b:visi.0000029664.99615.94.
6. Bay H., Ess A., Tuytelaars T., Van Gool L. SURF: Speeded Up Robust Features // Computer Vision and Image Understanding (CVIU). 2008. Vol. 110, №. 3. 14 p. DOI: 10.1016/j.cviu.2007.09.014.
7. Mikolajczyk K., Schmid C. A performance evaluation of local descriptors // IEEE Transactions on Pattern Analysis and Machine Intelligence. 2005. Vol. 10, №. 27. P. 1615-1630. DOI: 10.1109/tpami.2005.188.

8. Bouguet J. Y. Pyramidal implementation of the Lucas Kanade feature tracker // Intel Corporation, Microprocessor Research Labs. 2000. 9 p.
9. Muja M., Lowe D.G. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration // International Conference on Computer Vision Theory and Application VISSAPP'09. 2009. P. 331 340
10. Stepanov D.N., Tishchenko I.P., Poljakov A.V., Vatutin V.M., Sobolev D.B. Podsistema opredelenija polozhenija i orientacii bespilotnogo letatel'nogo apparata [The subsystem determine the position and orientation of the UAV]. 5 Vserossijskaja nauchno-tehnicheskaja konferencija "Aktual'nye problemy raketno-kosmicheskogo priborostroenija i informacionnyh tehnologij", OAO «Rossijskie kosmicheskie sistemy» (Moskva, 5–7 ijunja, 2012) [Proc. of 5th All-Russian Scientific Conference "Actual problems of missile and space instrumentation and information technology", Joint Stock Company "Russian Space Systems" (Moscow, 5–7 june, 2012)], Moscow, Radiotehnika, 2013, P. 9–27
11. Kamarainen J.-K., Kyrki V., Kälviäinen H. Invariance Properties of Gabor Filter Based Features - Overview and Applications // IEEE Transactions on Image Processing. 2006. Vol. 15, No. 5. 12 p. DOI: 10.1109/tip.2005.864174.
12. Stepanov D.N. Metody i algoritmy opredelenija polozhenija i orientacii bespilotnogo letatel'nogo apparata s primeneniem bortovyh videokamer [Methods and algorithms for determining the position and orientation UAV usage of on-board videocameras]. Programmnye produkty i sistemy (International Journal). 2014. Vol. 1, No. 1. URL: http://www.swsys.ru/index.php?page=article&id=3776 (accessed: 06.07.2015).
13. Stewenius H., Engels C., Nister D. Recent developments on direct relative orientation // ISPRS Journal of Photogrammetry and Remote Sensing. 2006. Vol. 60, Issue 4. P. 284–294. DOI: 10.1016/j.isprsjprs.2006.03.005.
14. Fischler M.A., Bolles R.C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography // Comm. Of the ACM 24. 1981. P. 381–395. DOI: 10.1145/358669.358692.
15. A. Kravcov, V. Vezhnevec. Obshhaja formulirovka zadachi vneshnej kalibrovki kamery [General formulation of the problem of extrinsic camera calibration] // Computer graphics and multimedia. 2003. Vol. 1, No 2. URL: http://cgm.computergraphics.ru/content/view/34 (accessed: 06.07.2015).
16. FLANN – Fast Library for Approximate Nearest Neighbors. URL: http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN (accessed: 06.07.2015).
17. Bradski G., Kaehler A. Learning OpenCV. O'Reilly Media. September 2008. 576 p.

Received July 8, 2015.

УДК 004.4'23, 004.4'24 DOI: 10.14529/cmse150403

РАСПАРАЛЛЕЛИВАНИЕ ТЕСТОВ NAS NPB ДЛЯ СОПРОЦЕССОРА INTEL XEON PHI НА ЯЗЫКЕ FORTRAN-DVMH[1]

В.Ф. Алексахин, В.А. Бахтин, О.Ф. Жукова, А.С. Колганов, В.А. Крюков, И.П. Островская, Н.В. Поддерюгина, М.Н. Притула, О.А. Савицкая

В статье анализируется эффективность выполнения тестов NAS из пакета NPB 3.3.1 (EP, MG, BT, SP, LU) на узлах кластеров различной архитектуры, использующих многоядерные универсальные процессоры, графические ускорители фирмы NVidia и сопроцессоры фирмы Intel. Сравниваются характеристики тестов, разработанных на высокоуровневом языке Fortran-DVMH (далее FDVMH), и их реализации на других языках. Исследуется влияние различных оптимизаций для FDVMH-версий тестов NAS, необходимых для их эффективной работы на сопроцессоре Intel Xeon Phi. Представлены результаты запусков тестов при одновременном использовании всех ядер центрального процессора, графического процессора и сопроцессора Intel Xeon Phi.

Ключевые слова: DVMH, высокоуровневый язык программирования, ускоритель, сопроцессор, графический процессор, тесты NAS, Фортран.

Введение

В последнее время все чаще стали появляться вычислительные кластеры, в узлах которых помимо универсальных многоядерных процессоров установлены ускорители разных типов. В основном, это графические процессоры компании NVidia и сопроцессоры Intel Xeon Phi. Так, в списке Top500 [1] самых высокопроизводительных суперкомпьютеров мира, объявленном в ноябре 2014 года, 75 машин имеют в своем составе ускорители, из них 50 машин имеют ускорители NVidia, 25 — Intel. В четырех машинах используется комбинация ускорителей NVidia и Intel Xeon Phi.

Данная тенденция заметно усложняет процесс программирования кластеров, так как требует освоения на достаточном уровне сразу нескольких моделей и языков программирования. Традиционным подходом можно назвать использование технологии MPI для разделения работы между узлами кластера, а затем технологий OpenMP, CUDA, OpenCL или OpenACC для загрузки ядер центрального и графического процессоров. Поэтому при разработке программы для суперкомпьютера приходится точно знать его архитектуру.

С целью упрощения программирования распределенных вычислительных систем были предложены высокоуровневые языки программирования, основанные на расширении директивами стандартных языков, такие, как HPF [2], Fortran-DVM [3], C-DVM [4]. Также были предложены модели программирования и соответствующие, основанные на директивах, расширения языков для возможности использования ускорителей, такие, как OpenACC [5] и OpenMP 4.0 [6].

[1] Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2015».

1. Обзор параллельных архитектур

В данной главе рассматриваются следующие архитектуры: центрального процессора на примере Intel Ivy Bridge-EP [7], сопроцессора Xeon Phi (MIC — Many Integrated Cores) [8] на примере Knights Corner и ГПУ Nvidia Kepler [9] на примере GK110.

1.1. Архитектура Ivy Bridge-EP

В наиболее сложной модификации данной архитектуры используются три блока, включающие четыре ядра ЦПУ и фрагмент кэш-памяти третьего уровня объемом 2,5 МБ на ядро. Сдвоенная кольцевая шина обеспечивает взаимодействие между блоками внутри чипа, а мультиплексоры позволяют довести команды до ядра, которому они адресованы. Внешняя шина QPI (Quick Path Interconnect), использующаяся для соединения процессоров между собой и с чипсетом, работает на скорости до 9,6 ГТ/с. Встроенный контроллер PCI Express обеспечивает работу 40 линий третьего поколения. В 12-ядерном чипе предусмотрено два контроллера памяти, каждый из которых поддерживает работу памяти до DDR3-1866 в двухканальном режиме.

В данной архитектуре присутствуют SSE (Streaming SIMD Extensions) регистры и AVX (Advanced Vector Extensions) регистры. SSE включает в архитектуру процессора восемь 128-битных регистров и набор инструкций, работающих со скалярными и упакованными типами данных. Преимущество в производительности достигается в том случае, когда необходимо произвести одну и ту же последовательность действий над разными данными. В таком случае блоком SSE осуществляется распараллеливание вычислительного процесса между данными. AVX предоставляет различные улучшения, новые инструкции и новую схему кодирования машинных кодов:

- ширина векторных регистров SIMD увеличивается со 128 до 256 бит. Существующие 128-битные SSE-инструкции используют младшую половину новых 256-битных регистров, не изменяя старшую часть. Для работы с данными регистрами добавлены новые 256-битные AVX-инструкции. В будущем возможно расширение векторных регистров SIMD до 512 или 1024 бит;
- для большинства новых инструкций отсутствуют требования к выравниванию операндов в памяти. Однако рекомендуется следить за выравниванием на размер операнда, во избежание значительного снижения производительности;
- набор AVX-инструкций содержит в себе аналоги 128-битных SSE-инструкций для вещественных чисел. При этом, в отличие от оригиналов, сохранение 128-битного результата будет обнулять старшую половину 256-битного регистра. 128-битные AVX-инструкции сохраняют прочие преимущества AVX, такие как новая схема кодирования, трехоперандный синтаксис и невыровненный доступ к памяти.

1.2. Архитектура Knights Corner

В сопроцессоре с данной архитектурой может быть интегрировано до 61 ядра x86 с большими 512-разрядными векторными модулями (содержащие 512-битные AVX-регистры), работающими на частоте более 1 ГГц и обеспечивающими скорость вычислений двойной точности более 1 TFlops. Они расположены на двухслотовой карте PCI Express со специальной прошивкой на базе Linux. Intel Xeon Phi включает до 16 Гбайт памяти GDDR5. Безусловно, таким образом сконструированные сопроцессоры не рассчитаны на обработку основных задач, с которыми справляются процессоры семейства

Xeon E. Они преуспевают в параллельных задачах, способных использовать большое количество ядер для максимального эффекта. Основные характеристики:

- архитектура x86 с поддержкой 64 бит, четыре потока на ядро и до 61 ядра на сопроцессор;
- 512-битные AVX-инструкции, 512 Кбайт кэша L2 на ядро (до 30,5 Мбайт на всю карту Xeon Phi);
- поддержка Linux (специальная версия для Phi), до 16 Гбайт памяти GDDR5 на карту.

Можно заметить, что даже у самой старшей модели Intel Xeon Phi гораздо меньше ядер, чем у обычного графического процессора. Но нельзя сравнивать ядро MIC с ядром CUDA в соотношении один к одному, так как одно ядро Intel Xeon Phi — это четырехпоточный модуль с 512-бит SIMD.

1.3. Архитектура GK110

Чип Kepler GK110 был разработан в первую очередь для пополнения модельного ряда Tesla. Его цель — стать самым быстрым параллельным микропроцессором в мире. Чип GK110 не только превышает по производительности чип предыдущего поколения Fermi, но и потребляет значительно меньше энергии. GK110 в максимальной конфигурации состоит из 15 потоковых мультипроцессоров, называемых SMX, и шести 64-битных контроллеров памяти.

Каждый потоковый мультипроцессор SMX содержит 192 cuda-ядра для операций одинарной точности, 64 cuda-ядра для операций двойной точности, 32 специальных функциональных блока и 32 блока для загрузки и сохранения данных, четыре планировщика. Для всех SMX на ГПУ предусмотрен один общий кэш второго уровня, размер которого составляет 1,5 МБ. Также у каждого SMX есть свой собственный кэш первого уровня размером 64 КБ.

Для того чтобы использовать большие мощности ГПУ, необходимо отобразить вычислительно независимые участки кода на группы независимых виртуальных нитей. Каждая группа будет исполнять одну и ту же команду над разными входными данными. Для управления данной группой виртуальных нитей необходимо объединить их в блоки фиксированного размера. Данные блоки в архитектуре CUDA называются cuda-блоками.

Такой блок имеет три измерения. Все блоки объединяются в решетку блоков, которая также может иметь три измерения. В конечном счете, каждый cuda-блок будет обработан каким-либо потоковым мультипроцессором, и каждой виртуальной нити будет сопоставлена физическая. Минимальной единицей параллелизма на ГПУ является warp — группа из 32 независимых нитей, которые исполняются физически параллельно и синхронно.

2. Обзор пакета тестов NAS

NAS Parallel Benchmarks [10] — комплекс тестов, позволяющий оценивать производительность суперкомпьютеров. Тесты разработаны и поддерживаются в NASA Advanced Super-computing (NAS) Division (ранее NASA Numerical Aerodynamic Simulation Program), расположенном в NASA Ames Research Center. Версия 3.3 пакета NPB включает в себя 11 тестов. В данной статье рассматривается распараллеливание на

ЦПУ и сопроцессоры в модели DVMH [11] тестов EP, MG, BT, LU, SP, для которых ранее были разработаны параллельные версии для кластеров с использованием языка Fortran-DVM, а также параллельные версии для графических процессоров с использованием языка Fortran-DVMH:

- MG (Multi Grid) — тест вычисляет приближенное решение трехмерного уравнения Пуассона («трехмерная решетка») в частных производных на сетке NxNxN с периодическими граничными условиями (функция на всей границе равна 0 за исключением заданных 20 точек). Размер сетки N определяется классом теста;
- EP (Embarrassingly Parallel) — чрезвычайно параллельный. Тест вычисляет интеграл методом Монте-Карло; предназначен для измерения первичной вычислительной производительности плавающей арифметики. Этот тест может быть полезен, если на кластере будут решаться задачи, связанные с применением метода Монте-Карло;
- BT (Block Tridiagonal) — блочная трехдиагональная схема. Тест решает синтетическую систему нелинейных дифференциальных уравнений в частных производных (трехмерная система уравнений Навье—Стокса для сжимаемой жидкости или газа), используя блочную трехдиагональную схему с методом переменных направлений;
- SP (Scalar Pentadiagonal) — скалярный пентадиагональный. Тест решает синтетическую систему нелинейных дифференциальных уравнений в частных производных (трехмерная система уравнений Навье—Стокса для сжимаемой жидкости или газа), используя скалярную пятидиагональную схему.
- LU (Lower-Upper) — разложение при помощи симметричного метода Гаусса—Зейделя. Тест решает синтетическую систему нелинейных дифференциальных уравнений в частных производных (трехмерная система уравнений Навье—Стокса для сжимаемой жидкости или газа), используя метод симметричной последовательной верхней релаксации.

3. Отображение программ на разные архитектуры в системе DVM

В 2011 году в Институте прикладной математики им. М.В. Келдыша РАН была разработана высокоуровневая модель программирования для поддержки кластеров с графическими ускорителями. Модель получила название DVMH [11] (DVM for Heterogeneous systems). Она является расширением модели DVM и позволяет с небольшими изменениями перевести DVM-программу для кластера в DVMH-программу для кластера с графическими ускорителями.

В 2014 году с появлением сопроцессоров Intel Xeon Phi возникла необходимость в их поддержке DVM-системой.

Всего существует три варианта выполнения программ на сопроцессоре Xeon Phi [12]:

- Native режим — программа компилируется и выполняется только на сопроцессоре;
- Offload режим — программа компилируется для ЦПУ, некоторые участки кода компилируются и для ЦПУ и для Xeon Phi. Данные участки кода помечаются специальными директивами OpenMP 4.0 (в общем случае #pragma offload).

  Программа выполняется так же, как и в случае использования ГПУ: последовательная часть — на ЦПУ, параллельная часть со специальными указаниями — на

сопроцессоре. При этом возникает та же проблема, что и при использовании ГПУ — загрузка и выгрузка данных в собственную память сопроцессора через PCIe. В случае отсутствия сопроцессора в узле, специальные участки кода будут выполнены на ЦПУ.

- Symmetric режим — одна и та же программа компилируется отдельно для ЦПУ и отдельно для сопроцессора. Скомпилированные программы одновременно запускаются на сопроцессоре и ЦПУ и могут синхронизироваться с помощью технологии MPI.

Основной режим выполнения, который был выбран для реализации в DVMH, — симметричный (symmetric). Данный режим позволяет настраивать баланс между ЦПУ и сопроцессором с помощью технологии MPI и использовать технологию OpenMP для лучшей загрузки ядер внутри каждого устройства. Также можно запустить FDVMH-программу только на сопроцессоре (native режим), используя, например, один MPI-процесс и максимальное количество OpenMP-нитей, поддерживаемое конкретным сопроцессором.

В итоге, распараллеливая программы с использованием модели DVMH, не нужно выбирать ту или иную архитектуру, будь то графические процессоры, центральные процессоры или сопроцессоры Intel Xeon Phi, так как эта модель поддерживает использование всех перечисленных архитектур как по отдельности, так и одновременно в рамках одной программы.

4. Реализация поддержки Xeon Phi в FDVMH-компиляторе

Вычислительным регионом в DVMH-программе называется часть программы с одним входом и одним выходом для возможного выполнения на одном или нескольких вычислительных устройствах. Регион может содержать несколько параллельных циклов. Данный участок программы помечается директивой REGION.

Регион может быть выполнен на одном или сразу нескольких устройствах: ускорителях, ЦПУ, сопроцессорах. При этом на ЦПУ или сопроцессоре может быть выполнен любой регион. Для выполнения региона на различных ускорителях на регион могут накладываться различные дополнительные ограничения. Например, на CUDA-устройстве может быть выполнен любой регион, в котором нет операторов ввода/вывода или вызовов внешних процедур.

Для каждого региона можно указать тип вычислителя, на котором следует его выполнять. Для этого предназначена спецификация TARGETS. Вложенные (статически или динамически) регионы не допускаются. DVM-массивы распределяются между выбранными вычислителями (с учетом весов и быстродействия вычислителей), нераспределенные данные размножаются. Витки параллельных DVM-циклов внутри региона делятся между выбранными для выполнения региона вычислителями в соответствии с правилом отображения параллельного цикла, заданного в директиве параллельного цикла.

4.1. Генерация обработчиков для параллельных циклов

Компилятор с языка Fortran-DVMH преобразует исходную программу в параллельную программу на языке Fortran с вызовами функций системы поддержки выполнения DVMH-программ (RTS, RunTime System). Кроме того, компилятор создает для каждой

исходной программы несколько дополнительных модулей для обеспечения выполнения регионов программы на ГПУ с использованием технологии CUDA и на ЦПУ и сопроцессоре с использованием технологии OpenMP.

Для каждого параллельного цикла из вычислительного региона компилятор генерирует процедуру-обработчик и ядро для вычислений на ГПУ, а также процедуру-обработчик для выполнения этого параллельного цикла на ЦПУ и сопроцессоре. Обработчик — это подпрограмма, осуществляющая обработку части параллельного цикла на конкретном вычислительном устройстве. Аргументами обработчика являются описатель устройства и часть параллельного цикла.

Обработчик запрашивает порцию для выполнения (границы цикла и шаг), конфигурацию параллельной обработки (количество нитей), инициализацию редукционных переменных и иную системную информацию у RTS. CUDA-обработчик для выполнения частей цикла вызывает специальным образом сгенерированное CUDA-ядро с параметрами, полученными во время выполнения от RTS. CUDA-ядро выполняется на графическом процессоре, производя вычисления, составляющие тело цикла. Для обработки частей цикла ЦПУ-обработчик использует технологию OpenMP для распределения вычислений. По умолчанию предполагается, что вычислительный регион может выполняться на архитектурах всех типов, которые присутствуют в узле кластера, и компилятор генерирует обработчики для ЦПУ, сопроцессора и CUDA-устройства.

Одной из основных причин замедления программ, выполняемых на ЦПУ или сопроцессоре, является линеаризация распределенных массивов, выполняемая компилятором Fortran-DVMH. Память для элементов таких массивов выделяется системой RTS. Для локальной секции массива на каждом процессоре память отводится в соответствии с форматом распределения данных и с учетом теневых граней. Все ссылки к элементам распределенных массивов вида ARRAY(I,J,K) заменяются компилятором на ссылки вида:

$$BASE(ARRAY\_OFFSET+I+C\_ARRAY1*J+C\_ARRAY2*K),$$

где BASE — это база, относительно которой адресуются все распределяемые массивы; ARRAY_OFFSET — смещение начала массива относительно базы; C_ARRAYi — коэффициенты адресации распределенного массива. Такая программа хуже распознается и оптимизируется стандартными Fortran компиляторами.

Для решения данной проблемы компилятор FDVMH по специальной опции может генерировать оптимизированные обработчики, чтобы распределенные массивы передавались в хост-обработчики как массивы, перенимающие размер (assumed shape arrays). Такой подход позволяет не преобразовывать обращения к массивам в линейную форму внутри хост-обработчиков и существенно ускорить выполнение программы на ЦПУ или сопроцессоре.

4.2. Использование клаузы collapse

В отличие от центральных процессоров, количество нитей на которых не превосходит 24 (для последних Intel Xeon E), сопроцессоры Intel Xeon Phi могут иметь 244 потока. Существует множество задач, в которых циклы не только одномерные, но и двухмерные и трехмерные. К таким задачам, в частности, относятся многие тесты из пакета NPB. Директива OpenMP *!$OMP PARALLEL FOR*, генерируемая компилятором FDVMH в хост-обработчиках, распространяется только на самый верхний цикл из всего

гнезда. Если количество итераций такого цикла меньше, чем количество доступных нитей, то производительность будет снижаться.

Для оптимизации работы хост-обработчиков на Xeon Phi необходимо генерировать дополнительную клаузу COLLAPSE в OpenMP-директиве. Клауза COLLAPSE позволяет распределить выполнение нескольких циклов гнезда, что позволяет задействовать все ядра сопроцессора. Параметром данной клаузы служит количество вложенных циклов, на которые она распространяется. Распространять данную клаузу на все вложенные циклы не эффективно, так как один и тот же поток будет обрабатывать элементы, не лежащие подряд, и в этом случае будет больше промахов в L2 кэш.

При статическом анализе во время компиляции DVMH-программы не удается определить количество вложенных циклов, на которое может быть распространена клауза COLLAPSE. Поэтому во время компиляции генерируется несколько хост-обработчиков, в каждом их которых расставляется клауза COLLAPSE(N), где $N = 1, 2, 3, .. m$, а m есть количество циклов в гнезде. В момент выполнения программы RTS определяет обработчик, который будет выполнять параллельный цикл.

При компиляции DVMH-программы пользователь может задать опцию компилятору *-collapseN*, где N — целое число, большее 0. Тогда для каждого параллельного цикла в хост-обработчике в OpenMP-директиву будет добавлена клауза COLLAPSE(N).

4.3. Балансировка нагрузки в DVM-системе

Одним из важных аспектов функционирования такой программной модели, как DVMH, является вопрос отображения исходной программы на все уровни параллелизма и разнородные вычислительные устройства. Важными задачами механизма отображения является обеспечение корректного выполнения всех поддерживаемых языком конструкций на разнородных вычислительных устройствах, балансировка нагрузки между вычислительными устройствами, а также выбор оптимального способа выполнения каждого участка кода на том или ином устройстве.

Параллелизм в DVMH-программах проявляется на нескольких уровнях:

- распределение данных и вычислений по MPI-процессам. Этот уровень задается директивами распределения и перераспределения данных и спецификациями параллельных подзадач и циклов. На данном уровне происходит отображение программы на узлы кластера. Внутри каждого узла может находиться несколько MPI-процессов, которые могут быть отображены на ядра ЦПУ или ядра сопроцессора;
- распределение данных и вычислений по вычислительным устройствам при входе в вычислительный регион;
- параллельная обработка в рамках конкретного вычислительного устройства. Этот уровень появляется при входе в параллельный цикл, находящийся внутри вычислительного региона. На данном уровне происходит отображение вычислений на OpenMP-нити и архитектуру CUDA.

Согласно данному разделению в DVMH-модели существует два уровня балансировки между вычислительными устройствами: задание веса каждого MPI-процесса, отображаемого на ядра ЦПУ или сопроцессора, и задание соотношения вычислительной мощности между ядрами ЦПУ и ГПУ для каждого MPI-процесса.

Для настройки производительности DVMH-программы не требуется ее перекомпиляция. Чтобы определить, как соотносятся веса MPI-процессов, пользователю необхо-

димо указать вектор весов в специальном файле с параметрами запуска DVMH-программы. В соответствии с указанными весами, RTS разделит данные между MPI-процессами во время работы программы. По умолчанию все MPI-процессы считаются одинаковыми.

Для балансировки нагрузки между ЦПУ и ГПУ существуют следующие механизмы. Все настройки осуществляются при помощи переменных окружения. Пользователь может определить количество нитей в терминах OpenMP на ЦПУ (или сопроцессоре), которое нужно использовать. Также можно включить или отключить выполнение параллельного цикла на одном или нескольких ГПУ. Для того, чтобы указать производительность ЦПУ и ГПУ, существуют две переменные окружения:

- DVMH_CPU_PERF — относительная производительность ЦПУ. Для разных MPI-процессов, выполняемых на ядрах ЦПУ или ядрах сопроцессора Intel Xeon Phi, существует возможность указать различные значения этого параметра;
- DVMH_CUDAS_PERF — относительная производительность ГПУ, задается списком вещественных чисел через пробел или запятую. Список является закольцованным. Это позволяет не расписывать производительности всех ГПУ.

Таким образом, DVM-система позволяет эффективно использовать все устройства различной архитектуры, установленные в узлах кластера, путем двухуровневой балансировки: определение веса каждого из MPI-процессов и определение соотношения производительности между OpenMP-нитями и CUDA устройствами внутри каждого из MPI-процессов.

5. Применение AVX-инструкций

Минимальной единицей параллелизма в терминах ГПУ является warp, который состоит из 32-х независимых нитей. Уже при написании параллельной программы с использованием программной модели CUDA прикладной программист сам определяет warp'ы, объединяет их в блоки, а затем в сетки блоков, тем самым указывая, где и что будет исполняться параллельно и векторно. Можно считать, что минимальный размер вектора в архитектуре ГПУ равен 32-м элементам. Дальнейшие сложности по оптимизации, планированию и загрузке/выгрузке данных берет на себя компилятор NVidia и аппаратура ГПУ.

Минимальной единицей параллелизма одного ядра ЦПУ или сопроцессора можно назвать векторный AVX-регистр, который может обрабатывать 256 или 512 бит данных за одну операцию соответственно. В данном случае прикладной программист работает сначала с последовательным кодом программы, а затем применяет высокоуровневые директивные расширения OpenMP. И в отличие от ГПУ, в программе, ориентированной на ЦПУ или сопроцессор, нет явных указаний о векторизации операций – всю работу по векторизации берет на себя компилятор. Поэтому есть два варианта написания эффективных параллельных программ, использующих векторные регистры AVX:

- использовать директивы компилятора или OpenMP (например, «omp simd»);
- использовать низкоуровневые команды или AVX-инструкции (на уровне intrinsic-функций).

Первый способ не всегда реализуем, так как компилятор не всегда может справиться с векторизацией, даже если она возможна. Рассмотрим два варианта одного и того же цикла (см. рис. 1).

```
#if VER1 /*** первый вариант цикла ***/
  double rhs[5][162][162][162], u[5][162][162][162];
  #define rhs(m,i,j,k) rhs[(m)][(i)][(j)][(k)]
  #define u(m,i,j,k)     u[(m)][(i)][(j)][(k)]
#endif
#if VER2 /*** второй вариант цикла ***/
     double rhs[162][162][162][5], u[162][162][162][5];
     #define rhs(m,i,j,k) rhs[(i)][(j)][(k)][(m)]
     #define u(m,i,j,k)     u[(i)][(j)][(k)][(m)]
#endif
#pragma omp parallel for
for(int i = 0; i < Ni; i++)
{
     for(int j = 0; j < Nj; j++)
     {
        for(int k = 0; i < Nk; k++)
        { /*** первая группа операторов ***/
          rhs(0, i, j, k) = F(u(0, i, j, k), u(0, i+1, j, k));
          rhs(1, i, j, k) = F(u(1, i, j, k), u(1, i-2, j, k));
          rhs(2, i, j, k) = F(u(2, i, j, k), u(2, i, j+1, k));
          rhs(3, i, j, k) = F(u(3, i, j, k), u(3, i, j-1, k));
          rhs(4, i, j, k) = F(u(4, i, j, k), u(4, i+1, j, k+1));
          /*** вторая группа операторов ***/
               for (int m = 0; m < 5; m++)
                rhs(m, i, j, k) += F(u(m, i, j, k), u(m, i, j+1, k));
        }
     }
}
```

Рис. 1. Два варианта параллельного цикла

Данные циклы производят одинаковые вычисления, но отличаются расположением данных в памяти. В первом варианте (VER1) компилятор не может векторизовать ни одну группу операторов, потому что самое быстро индексируемое измерение является измерением параллельного цикла. Во втором варианте (VER2) компилятор успешно векторизует вторую группу операторов, так как самое быстро индексируемое измерение не является измерением параллельного цикла и данные в памяти по этому измерению расположены подряд.

Тем не менее, бывают ситуации, когда выгодно использовать расположение данных такое, как в первом варианте (VER1, рис. 1). Одна из таких ситуаций — перестановка массивов в предшествующем параллельном цикле программы для лучшей локализации данных. И для того, чтобы не произошло замедления выполнения рассматриваемого цикла, необходимо использовать векторные AVX-инструкции непосредственно в коде программы.

Для первого варианта (VER1) возможна векторизация двух групп операторов, потому что данные в памяти по самому быстрому измерению параллельного цикла лежат подряд. Пример преобразования одного из операторов с применением AVX-инструкций показан на рис. 2.

```
double rhs[5][162][162][162], u[5][162][162][162];
#pragma omp parallel for
for(int i = 0; i < Ni; i++)
{
     for(int j = 0; j < Nj; j++)
     {
     /*** изменение индексного пространства цикла ***/
        for(int k = 0; i < Nk; k+=8)
        { /*** rhs[0][i][j][k] = u[0][i][j][k] + u[0][i+1][j][k]; ***/
          _mm512_store_pd(&rhs[0][i][j][k],
               _mm512_add_pd(
                _mm512_load_pd(u[0][i][j][k]),
                _mm512_load_pd(u[0][i+1][j][k]
               )
          );
        }
     }
}
```

Рис. 2. AVX-преобразование

За одну операцию 512-битная AVX-команда может обрабатывать 8 элементов типа double. Использование данного подхода позволяет ускорить программу примерно в 8 раз. Для подтверждения данного факта была реализована одна из вычислительно сложных процедур теста NPB SP на языке C++ с использованием AVX-инструкций и без. Полученные программы компилировались Intel-компилятором с флагом оптимизации -O3.

В результате проведенного эксперимента было достигнуто ускорение порядка 6,3 раз по сравнению с той же процедурой без использования AVX-инструкций (см. табл. 1), что говорит о высокой эффективности использования данного подхода. Возможность использования AVX-инструкций в Fortran-DVMH компиляторе является предметом для дальнейших исследований.

Таблица 1

Сравнение времени выполнения разных реализаций процедуры compute_rhs теста SP

Класс	Xeon Phi 240th (c)	Xeon Phi 240th + AVX512 (c)	Ускорение
A	2,9	1,8	1,55
B	13,4	4,8	2,76
C	118,1	18,7	6,31

6. Полученные результаты. Сравнение с MPI и OpenMP версиями тестов NASA

Для оценки эффективности распараллеливания программ с использованием модели DVMH было произведено сравнение времени выполнения тестов NAS (MG, EP, SP, BT и LU из пакета NPB 3.3), написанных на языке Fortran-DVMH, с временем выполнения стандартных версий этих тестов, использующих технологии MPI и OpenMP. Для каждого теста имеется всего один вариант параллельной FDVMH-программы, который может быть скомпилирован под каждую из архитектур: ЦПУ, сопроцессор или ГПУ. Все оптимизации, которые были проделаны над данными тестами, были описаны в статье [13].

Тестирование производилось на сервере с установленными на нем 6-ядерным (12-поточным) процессором Intel Xeon E5-1660v2 с 24 Гб оперативной памяти типа DDR3, сопроцессором Intel Xeon Phi 5110P с 8 Гб оперативной памяти типа GDDR5 и ГПУ NVidia GTX Titan с 6 Гб оперативной памяти типа GDDR5. Основные результаты представлены в табл. 2.

Таблица 2

Времена выполнения тестов NASA различных версий (в секундах)

Класс	Xeon E5 (4-12th)				Xeon Phi (64-240th)			GTX Titan
	Serial	MPI	OpenMP	DVMH	MPI	OpenMP	DVMH	DVMH
BT								
A	40,7	12,13	10,2	6,97	11,08	11,5	7,68	2,84
B	166,9	54,9	43,07	28,8	33,1	32,15	20,9	9,16
C	713,3	223,1	176,7	118,1	119,4	105,7	74	31,05
SP								
A	28,6	17,8	14,6	6,75	12,03	13,3	11,6	2,4
B	116,94	96,8	57,1	29,18	33,4	38,7	27	10,2
C	483,24	408,6	425,2	122,2	124,03	128,2	120	31
LU								
A	35,07	9,6	8,31	4,7	15,05	16,5	16,5	4,18
B	148,56	35,2	31,1	22,7	47,01	44,5	57,7	11,69
C	852,3	291,4	351,9	99	162,4	134,2	157,3	33,73
MG								
A	1,06	0,57	0,7	0,38	0,36	0,22	0,61	0,13
B	4,96	2,7	3,22	2,14	1,7	1,13	2,8	0,58
C	42,3	25,7	34,6	16,2	10,9	6,39	15,5	3,36
EP								
A	16,73	1,63	1,76	1,5	0,94	0,89	0,78	0,48
B	67,33	6,6	7,03	5,99	3,94	3,31	2,99	1,17
C	266,3	26,1	26,3	23,96	14,8	13,31	11,6	4,27

На рис. 3 показано ускорение теста EP, написанного с использованием языка FDVMH, по сравнению с последовательной версией данной программы, выполненной на одном ядре ЦПУ. Данный тест выполнялся на разных архитектурах по отдельности, а также в следующих комбинациях: ЦПУ + ГПУ, ЦПУ + сопроцессор и сопроцессор + ЦПУ + ГПУ. На рис. 3 для каждой конфигурации запуска указывается количество MPI-процессов и количество OpenMP-нитей внутри каждого из процессов. Красным и сиреневым цветом показаны случаи, когда дополнительно использовалась балансировка нагрузки путем задания соотношения весов всех ядер ЦПУ и ГПУ и соотношения весов MPI-процессов, отображаемых на ЦПУ и сопроцессор.

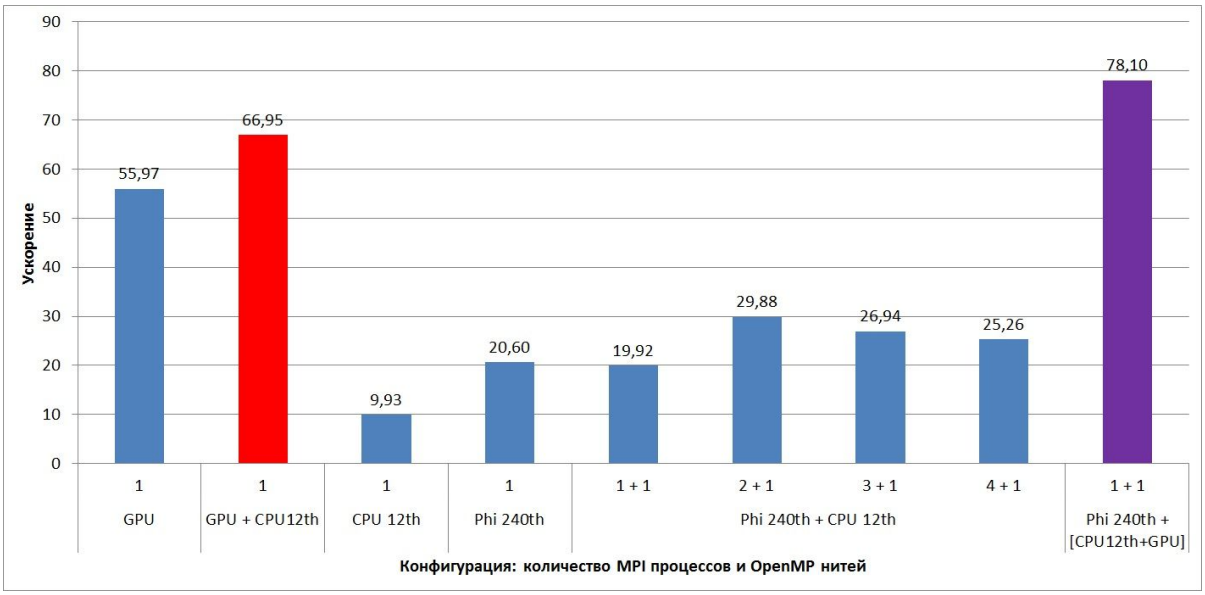


Рис. 3. Использование балансировки нагрузки в DVMH-программе на примере теста EP класса C

В итоге, на данном тесте при одновременном использовании ГПУ и ЦПУ удалось достичь производительности, которая на 17 % больше производительности выполнения теста на одном ГПУ. При этом соотношение производительности ЦПУ и ГПУ было установлено как 1 : 5,7 соответственно.

При совместном использовании сопроцессора и ЦПУ удалось достичь производительности, которая на 30 % больше производительности выполнения теста на одном сопроцессоре. Наиболее выгодное соотношение MPI-процессов следующее: два MPI-процесса на Xeon Phi и один MPI-процесс на Xeon E5, при этом MPI-процессы имеют одинаковый вес, либо по одному MPI-процессу на ЦПУ и сопроцессор, и веса процессов соотносятся как 1 : 2 соответственно.

При использовании всех устройств, установленных в узле, удалось достичь производительности, которая на 28 % больше производительности выполнения теста EP на ГПУ и в 3,7 раз превышает производительность выполнения этого теста на сопроцессоре.

Заключение

В данной работе исследовано расширение DVM-системы для поддержки ускорителей Intel Xeon Phi. Реализация данного расширения — существенный шаг в направлении обеспечения эффективной переносимости FDVMH-программ, когда одна и та же параллельная программа может эффективно выполняться на кластерах различной архитектуры, использующих многоядерные универсальные процессоры, графические ускорители и сопроцессоры Intel Xeon Phi. Реализованное отображение DVMH-программы на ЦПУ и сопроцессор позволяет применять многие из тех оптимизаций последовательной программы, которые были предложены в статье [14], и позволяющие эффективно отображать данные программы на графический процессор.

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 13-07-00580 а.

Литература

1. Top500 List — November 2014. URL: http://top500.org/list/2014/11/ (дата обращения 01.04.2015).
2. High Performance Fortran. URL: http://hpff.rice.edu (дата обращения 01.04.2015).
3. Коновалов, Н.А. Fortran DVM — язык разработки мобильных параллельных программ / Н.А. Коновалов, В.А. Крюков, С.Н. Михайлов, А.А. Погребцов // Программирование. — 1995. — № 1. — С. 49–54.
4. Коновалов, Н.А. C-DVM — язык разработки мобильных параллельных программ / Н.А. Коновалов, В.А. Крюков, Ю.Л. Сазанов // Программирование. — 1999. — № 1. — С. 54–65.
5. OpenACC. URL: http://www.openacc-standard.org/ (дата обращения 01.04.2015).
6. OpenMP 4.0 Specifications. URL: http://openmp.org/wp/openmp-specifications/ (дата обращения 01.04.2015).
7. Архитектура Intel Ivy Bridge-EP. URL: http://www.intel.ru/content/www/ru/ru/secure/intelligent-systems/privileged/ivy-bridge-ep/xeon-e5-1600-2600-v2-bsdl.html (дата обращения 30.11.2014).
8. Архитектура Intel MIC. URL: https://software.intel.com/mic-developer (дата обращения 01.04.2015).
9. Архитектура Nvidia Kepler. URL: http://www.nvidia.com/content/PDF/kepler/NVIDIA-kepler-GK110-Architecture-Whitepaper.pdf (дата обращения 01.04.2015).
10. NAS Parallel Benchmarks. URL: http://www.nas.nasa.gov/publications/npb.html (дата обращения 01.04.2015).
11. Бахтин, В.А. Расширение DVM-модели параллельного программирования для кластеров с гетерогенными узлами / В.А. Бахтин, М.С. Клинов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, Ю.Л. Сазанов // Вестник Южно-Уральского государственного университета. Серия: «Математическое моделирование и программирование». — 2012. — № 18(277). — С. 82–92.
12. Intel Xeon Phi programming environment URL: https://software.intel.com/en-us/articles/intel-xeon-phi-programming-environment (дата обращения 01.04.2015).
13. Алексахин, В.Ф. Распараллеливание на графические процессоры тестов NAS NPB3.3.1 на языке Fortran DVMH / В.Ф. Алексахин, В.А. Бахтин, О.Ф. Жукова, А.С. Колганов, В.А. Крюков, Н.В. Поддерюгина, М.Н. Притула, О.А. Савицкая, А.В. Шуберт // Вестник Уфимского государственного авиационного технического университета.— 2015. — Т. 19, № 1(67). — С. 240–250.
14. Ramachandran, A. Performance Evaluation of NAS Parallel Benchmarks on Intel Xeon Phi / A. Ramachandran, J. Vienne, R. Wijngaart, L. Koesterke, I. Sharapov // In Proceedings of the 42nd International Conference on Parallel Processing. — 2013. — P. 736–743. DOI: 10.1109/icpp.2013.87.

Алексахин Валерий Федорович, старший научный сотрудник, ФГБУН Институт прикладной математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), valex@keldysh.ru.

Бахтин Владимир Александрович, к.ф.-м.н., заведующий сектором, ФГБУН Институт прикладной математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), bakhtin@keldysh.ru.

Жукова Ольга Федоровна, научный сотрудник, ФГБУН Институт Прикладной Математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), socol@keldysh.ru.

Колганов Александр Сергеевич, младший научный сотрудник, ФГБУН Институт прикладной математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), 79854210702@ya.ru.

Крюков Виктор Алексеевич, д.ф.-м.н., заведующий отделом, ФГБУН Институт прикладной математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), krukov@keldysh.ru.

Островская Ирина Петровна, научный сотрудник, ФГБУН Институт прикладной математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), iostrov@keldysh.ru.

Поддерюгина Наталия Викторовна, к.ф.-м.н., старший научный сотрудник, ФГБУН Институт прикладной математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), konov@keldysh.ru.

Притула Михаил Николаевич, к.ф.-м.н., старший научный сотрудник, ФГБУН Институт прикладной математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), pritula@keldysh.ru.

Савицкая Ольга Антониевна, научный сотрудник, ФГБУН Институт прикладной математики им. М.В. Келдыша Российской академии наук (Москва, Российская Федерация), savol@keldysh.ru.

Поступила в редакцию 10 апреля 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 4, pp. 48–63

DOI: 10.14529/cmse150403

PARALLELIZATION OF NAS PARALLEL BENCHMARKS FOR INTEL XEON PHI COPROCESSOR IN FORTRAN-DVMH LANGUAGE

V.F. Aleksahin, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) valex@keldysh.ru,
V.A. Bakhtin, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) bakhtin@keldysh.ru,
O.F. Zhukova, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) socol@keldysh.ru,
A.S. Kolganov, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) 79854210702@ya.ru,
V.A. Krukov, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) krukov@keldysh.ru,
I.P. Ostrovskaya, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) iostrov@keldysh.ru,
N.V. Podderugina, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) konov@keldysh.ru,
M.N. Pritula, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) pritula@keldysh.ru,
O.A. Savitskaya, Keldysh Institute of Applied Mathematics Russian Academy of Sciences (Moscow, Russian Federation) savol@keldysh.ru

The article analyzes the effectiveness of the implementation of NAS benchmarks from NPB 3.3.1 package (EP, MG, BT, SP, LU) on cluster nodes with different architectures using multicore processors, NVidia graphics accelerators and Intel coprocessors. Characteristics of tests developed in high-level Fortran-DVMH language (hereafter referred to as FDVMH), and their implementation in other languages are compared. We research the effect of different optimization methods for FDVMH NAS benchmarks necessary for their effective work on Intel Xeon Phi coprocessor. The results of the simultaneous using of all cores of CPU, GPU and Intel Xeon Phi coprocessor are presented.

Keywords: DVMH, high-level programming language, accelerator, coprocessor, GPU, NAS Parallel Benchmarks, Fortran.

References

1. Top500 List — November 2014. URL: http://top500.org/list/2014/11/ (accessed: 01.04.2015).
2. High Performance Fortran. URL: http://hpff.rice.edu (accessed: 01.04.2015).

3. Konovalov N.A., Kryukov V.A, Mikhajlov S.N., Pogrebtsov A.A. Fortan DVM: a Language for Portable Parallel Program Development // Programming and Computer Software. 1995. Vol. 21, No. 1. P. 35–38.
4. Konovalov N.A., Krukov V.A, Sazanov Y.L. C-DVM — a Language for the Development of Portable Parallel Programs // Programming and Computer Software. 1999. Vol. 25, No. 1. P. 46-55.
5. OpenACC. URL: http://www.openacc-standard.org/ (accessed: 01.04.2015).
6. OpenMP 4.0 Specifications. URL: http://openmp.org/wp/openmp-specifications/ (accessed: 01.04.2015).
7. Intel Ivy Bridge-EP architecture.
URL: http://www.intel.ru/content/www/ru/ru/secure/intelligent-systems/privileged/ivy-bridge-ep/xeon-e5-1600-2600-v2-bsdl.html (accessed: 30.11.2014).
8. Intel MIC architecture. URL: https://software.intel.com/mic-developer (accessed: 01.04.2015).
9. Nidia Kepler architecture. URL: http://www.nvidia.com/content/PDF/kepler/NVIDIA-kepler-GK110-Architecture-Whitepaper.pdf (accessed: 01.04.2015).
10. NAS Parallel Benchmarks. URL: http://www.nas.nasa.gov/publications/npb.html (accessed: 01.04.2015).
11. Bakhtin V.A, Klinov M.S., Krukov V.A., Podderugina N.V., Pritula M.N., Sazanov Yu.L. Rasshirenie DVM-modeli parallel'nogo programmirovaniya dlya klasterov s geterogennymi uzlami [Extension of the DVM-model of parallel programming for clusters with heterogeneous nodes]. 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). P. 82–92.
12. Intel Xeon Phi programming environment. URL: https://software.intel.com/en-us/articles/intel-xeon-phi-programming-environment (accessed: 01.04.2015).
13. Aleksahin V.F, Bakhtin V.A, Zhukova O.F., Kolganov A.S., Krukov V.A., Podderugina N.V., Pritula M.N., Savitskaya O.A., Shubert A.V. Rasparallelivanie na graficheskie processory testov NAS NPB3.3.1 na jazyke Fortran DVMH [Parallelization on GPUs of NPB 3.3 NAS tests on Fortran DVMH language]. Vestnik Ufimskogo gosudarstvennogo aviacionnogo tehnicheskogo universiteta [Bulletin of Ufa State Aviation Technical University]. 2015, Vol. 19, No. 1(67). P. 240–250.
14. Ramachandran, A. Performance Evaluation of NAS Parallel Benchmarks on Intel Xeon Phi / A. Ramachandran, J. Vienne, R. Wijngaart, L. Koesterke, I. Sharapov // In Proceedings of the 42nd International Conference on Parallel Processing. — 2013. — P. 736–743. DOI: 10.1109/icpp.2013.87.

Received April 10, 2015.

УДК 004.7 DOI: 10.14529/cmse150404

РАЗРАБОТКА АЛГОРИТМИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ СИНТЕЗА ТОПОЛОГИЧЕСКИХ СТРУКТУР ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ

А.А. Сорокин, П.С. Резников

В работе показано, что одна из проблем систем телекоммуникаций — это неопределенность отношения элемента к определенной подсистеме сети передачи информации, элемент может использоваться как абонентское оборудование и как подсистема управления. В результате затрудняется внедрение ad hoc сетей в структуру систем мобильной связи. Для устранения проблемы разработан метод описания структуры системы инфокоммуникаций. Метод основан на представлении системы как единой логической структуры. Основа структуры — это виртуальные узлы. Совокупность виртуальных узлов образует виртуальную сеть. Пропускная способность виртуальной сети изменяется по вероятностным законам. Вероятностные законы определяются на основе статистики посещения виртуального узла физическими узлами. Область использования для систем связи заключается в повышении эффективности управления за счет прогнозирования альтернативных маршрутов передачи информации и синтеза систем связи различных стандартов. Результаты можно использовать для исследования взаимодействий социальных и экономических систем региона. Для демонстрации результатов проведен эксперимент по прогнозированию количества физических узлов в заданном виртуальном узле. По полученным данным рассчитана вероятность появления в виртуальном узле физических объектов. Результаты расчета позволяют построить карту посещаемости территории физическими узлами и рассчитать маршруты передачи трафика между виртуальными узлами.

Ключевые слова: система, анализ, инфокоммуникации, топология, структура, синтез, информация, управление, обработка, сеть.

Введение

Системы инфокоммуникаций интегрируются в разные виды деятельности человека. Область использования систем инфокоммуникаций достаточно широка, начиная от обеспечения голосовой связи и досуга и заканчивая участием в процессах управления транснациональными корпорациями. Учитывая факторы взаимного влияния социума и систем передачи и обработки информации, порождаются новые рыночные ниши и технические решения. Одна из общесистемных черт сетей инфокоммуникаций — взаимная интеграция (или конвергенция) различных программных и аппаратных средств передачи и обработки информации. В результате конвергенции технических и программных решений разрабатываются новые методы построения телекоммуникационных систем. Пример подобной конвергенции — создание класса ad hoc сетей. Согласно работам [1–4] термин ad hoc сети означает самоорганизующиеся распределенные системы передачи и обработки информации (СРСПОИ). В состав класса ad hoc сетей входят MANET (Mobile ad hoc networks) [1], VANET (Vehicular ad hoc networks) [2], Mesh и сенсорные сети [3, 4]. В результате взаимодействия структурных элементов подобных систем происходит синтез необходимых топологических структур в зависимости от внешних условий. Функционирование СРСПОИ происходит как без подключения к системам связи общего пользования, так и совместно с ними при помощи взаимодействия через шлюзы. Область использования СРСПОИ разнообразна — это обмен информацией между неболь-

шими группами подвижных и статичных пользователей (MANET сети); обмен служебной информацией между подвижными объектами с целью автоматизации управления транспортными потоками (VANET сети); быстро-развертываемые системы доступа к сети Интернет (Mesh сети); системы сбора данных с технологических объектов (сенсорные сети).

Широкое применение самоорганизующихся систем передачи и обработки информации сдерживаются относительно низким качеством передачи трафика и сложностями взаимодействия с системами связи общего пользования, например сетями мобильной связи [1–5]. Исследование способов повышения эффективности передачи трафика в системах связи с изменяемой топологией показали, что первопричиной ограничений являются принципы формального представления топологии систем данного класса, так как данные принципы применяются во время разработки протоколов, работа которых влияет на качество передачи информации [5, 6]. Следовательно, необходимо совершенствование формального аппарата, на основе которого разрабатываются протоколы управления самоорганизующихся систем передачи и обработки информации. Учитывая тенденцию на интеграцию структур ССПОИ с сетями связи общего пользования формальный аппарат должен предусматривать структурно-параметрический синтез двух классов систем и позволять прогнозировать развитие системы с учетом различных внешних воздействий.

Цель работы: разработка и программная реализация математического обеспечения для описания синтеза топологических структур систем инфокоммуникаций.

Статья состоит из следующих разделов. В первом разделе рассмотрены общие принципы формирования структур инфокоммуникационных систем, проанализированы тенденции их развития. Во втором разделе показаны ограничения современных моделей и методы описания топологии систем инфокоммуникаций. В третьем разделе описан новый метод формирования топологии сети инфокоммуникационной системы. В четвертом разделе описана реализация алгоритмического обеспечения, которая позволяет сформировать топологическую структуру из узлов различной степени подвижности. В заключении приводятся в обобщенном виде основные результаты работы.

1. Анализ структурных элементов инфокоммуникационных систем

Тенденции развития современного рынка инфокоммуникаций показывают эффект взаимного влияния социально-экономических и технических составляющих. К числу подобных составляющих можно отнести: спрос пользователей и предложения создателей контента, разработчиков программного обеспечения, различные виды абонентского оборудования, уровень развития сетей операторов связи [1, 3]. Примером подобного влияния стало появление мобильных устройств (ноутбуков и коммуникаторов) и развитие систем мобильной связи второго и четвертого поколений. Мобильные устройства вызвали потребность в доступе с них к сети Internet, что привело к интенсивному внедрению сетей мобильной связи третьего поколения с тарифными планами и средствами доступа к сети Internet. Доступные тарифные планы увеличили спрос и предложение смартфонов и планшетных компьютеров, создание Internet площадок для установки программ (Apple Store, Google play и др.), социальных сетей, видеохостингов. Возможностей сетей тертьего поколения стало недостаточно. Операторы стали внедрять сети четвертого по-

коления, начались изыскания в области систем связи новых поколений. Таким образом, на развитие оператора связи оказывает влияние поведение абонентов сети, а новые возможности сети порождают новые потребности у пользователей сети.

Принципы функционирования системы инфокоммуникаций описаны в концептуальных моделях. К числу подобных моделей относятся: модель взаимодействия открытых систем (OSI), модель стека протоколов TCP/IP, четырехуровневая модель управления сетью TMN (Telecommunication Management Network) [7]. Модели TCP/IP и OSI рассматривают систему с позиции решения задач передачи трафика. Модель TMN ориентирована менеджмент в управлении оператором связи, но не содержит, в явном виде, описание средств построения обратных связей между прогнозированием спроса пользователей сети и принятием решений в области управления сетью.

Опыт управления инфокоммуникационными бизнес-процессами показывает, что успех проекта зависит от правильного таргетинга абонентов. Таргетинг, согласно [8], — это выделение целевой аудитории по заданным критериям для демонстрации предложений. В рамках исследований под таргетингом абонентов понимается, выделение целевой группы, в зависимости от времени суток, места нахождения, социального статуса для предоставления желаемой инфокоммуникационной услуги. Поддержка и принятие решений по управлению бизнес-процессом в области инфокоммуникаций должна опираться на систему четко выстроенных обратных связей. Обратные связи должны передавать информацию о особенностях распределения абонентской базы сети инфокоммуникаций.

С учетом работы [9] систему инфокоммуникаций можно представить в виде совокупности множеств структурных элементов четырех подсистем:

- множества элементов абонентского оборудования — $\{Sb_\alpha\}$, где $\alpha \in [0,\mathrm{A}) : \mathrm{A} \to \infty$ число элементов множества абонентского оборудования,
- множества элементов подсистемы доступа — $\{Ap_\beta\}$, где $\beta \in [0,\mathrm{B}] : \mathrm{B} \neq \infty$ число элементов множества подсистемы доступа,
- множества элементов подсистемы транспортной сети — $\{Tn_\gamma\}$, где $\gamma \in [0,\Gamma] : \Gamma \neq \infty$ число элементов множества подсистемы транспортной сети,
- множества элементов подсистемы управления — $\{M_\phi\}$, где $\phi \in [0,\Phi] : \Phi \neq \infty$ число элементов множества подсистемы управления.

Объединение множеств $\{Sb_\alpha\}$, $\{Ap_\beta\}$, $\{Tn_\gamma\}$, $\{M_\phi\}$ образует топологическую структуру инфокоммуникационной системы:

$$IS \supseteq \{Sb_\alpha\} \cup \{Ap_\beta\} \cup \{Tn_\gamma\} \cup \{M_\phi\}. \qquad (1)$$

Обзор работ [1–3, 9, 10] и практический опыт эксплуатации систем инфокоммуникаций показывает, что большая часть оборудования всех подсистем — это программно-аппаратные комплексы. Принципы работы аппаратной части оборудования различных подсистем идентичны. Различия оборудования, как правило, заключаются в функциональных особенностях отдельных модулей. Например, мощности передающего оборудования или объемом встроенной памяти. Основные отличия оборудования различных подсистем наблюдаются в области программного обеспечения (ПО). В зависимости от вида ПО устройство выполняет разные функции: обрабатывает приложения пользователей, решает задачи поиска маршрутов передачи информации. Принципы функционирования ПО разного оборудования могут совпадать, так как реализованы на основе идентичных операционных систем (ОС), например на основе Unix подобных ОС (Linux, Free-

BSD) или Windows ОС. Общий признак систем инфокоммуникаций — возможность использовать один программно-аппаратный комплекс в разных подсистемах. Подобная возможность позволяет оборудованию одной подсистемы решать задачи других подсистем, в зависимости от состояния сети. В результате нарушается распределение функций между подсистемами абонентского оборудования, уровня доступа, транспортной сети и управления. В рамках проводимых исследований подобное явление целесообразно назвать эффектом перераспределения функциональных задач. Эффект перераспределения функциональных задач оборудования наблюдается в сети Internet. В сети Internet устройство может одновременно выполнять функции сервера (подсистема управления), рабочей станции (подсистема абонентского оборудования) и маршрутизатора (подсистема управления и транспортной сети). Практические примеры реализации — это сети peer-to-peer, различные виды VPN (Virtual Private Network).

Явное распределение функций наблюдалось в телефонных сетях общего пользования, системах сотовой связи первого и второго поколений. Явное распределение функций между подсистемами инфокоммуникационной сети можно описать соотношением вида:

$$Sb \Leftrightarrow \{F_{Sb}\} \wedge Ap \Leftrightarrow \{F_{Ap}\} \wedge Tn \Leftrightarrow \{F_{Tn}\} \wedge M \Leftrightarrow \{F_M\}, \quad (2)$$

где $\{F_{Sb}\} \wedge \{F_{Ap}\} \wedge \{F_{Tn}\} \wedge \{F_M\}$ — множество функциональных задач, которые решают элементы подсистем абонентского оборудования, доступа, транспортной сети, управления. В сети Internet перераспределение функциональности оборудования наблюдается со стороны подсистемы доступа, что можно описать соотношением вида:

$$Sb = \{F_{Sb}\} \cup \{f_M\} : \{f_M\} \subset \{F_M\}, \quad (3)$$

где $\{f_M\}$ — это часть функций оборудования подсистемы управления. Обзор ряда зарубежных [1, 2, 11] и отечественных [5, 12] работ показывает, что в самоорганизующихся системах передачи и обработки информации, перераспределение функциональности представляется в виде:

$$Sb = \{F_{Sb}\} \cup \{f_{Ap}\} \cup \{f_{Tn}\} \cup \{f_M\} : \{f_{Ap}\} \subset \{F_{Ap}\} \wedge \{f_{Tn}\} \subset \{F_{Tn}\} \wedge \{f_M\} \subset \{F_M\}, \quad (4)$$

где $\{f_{Ap}\} \wedge \{f_{Tn}\}$ — это часть функций оборудования подсистемы доступа и транспортной сети.

Обобщение соотношений (2–4) показывает процесс эволюции принципов построения и управления сетями инфокоммуникаций, а функционал (1) — принцип объединения задач управления по передаче информации в телекоммуникационной системе. Из теории управления известно, что для решения оптимизационных задач применяются модели объекта. «Классические» модели сетей связи развивались на основе принципа описанного функционалом (2). Для современных сетей, сформированных с учетом функционала (3), классические модели модернизировались для решения частных задач. Основная проблема общей методологии моделирования системы телекоммуникаций, сформированной с учетом функционалов (3 и 4) — неопределенность отношения элемента к определенной подсистеме сети передачи информации, что приводит к увеличению затрат времени на прогнозирование развития системы. Для устранения проблемы необходим поиск нового метода описания структуры сети. Новый метод должен описывать топологию в виде единой структуры с узлами различной степени подвижности и типом управления. Дополнительно метод должен показывать взаимодействие с социальными и экономическими процессами, которые затрагивают абонентов.

2. Анализ методов формального описания топологической структуры

Методы формального описания структур инфокоммуникационных систем базируются на теории графов, согласно данной теории, сеть представляется в виде графа [9]:

$$G = (V, E), \tag{5}$$

где V — множество вершин $\{v_i\} : i \in [2, \mathrm{I}] : \mathrm{I} \to \infty$, различное оборудование связи: абонентские терминалы, точки доступа, базовые станции, серверы, маршрутизаторы, коммутаторы и др. оборудование, которое создает поток информационных данных или канализирует его. E — множество ребер графа $\{e_j\} : j \in [1, \Psi] : \Psi \to \infty$ — кабельные или беспроводные каналы связи. При этом, $\forall v_i \wedge \forall e_j \ \exists M$, где M — вес вершины или ребра графа сети, которое имеет численное значение. Данное значение описывает параметры элемента сети, если $M = 0 \vee \infty$, значит произошла аварийная ситуация или между узлами отсутствует канал связи (в некоторых отдельных случаях $M = 0$, означает, что вершина «замкнута на себя», что изначально оговаривается во время использования модели). При помощи теории графов описаны системы мобильной и проводной связи, системы космической связи, различные классы самоорганизующихся систем передачи и обработки информации. Для каждого класса систем разработаны модели. Различия моделей обусловлены особенностями топологической структуры системы. Ad hoc сети описывается моделями, основанными на теории случайных графов [6]. Системы мобильной связи описываются разновидностями классической графовой модели, которые дополняются методами теории вероятности [11]. Часто для одного класса систем, в зависимости от вида моделируемой подсистемы, используются различные виды моделей. Например, для систем сотовой связи для описания топологической структуры элементов множеств $\{Ap_\beta\}$, $\{Tn_\gamma\}$ и $\{M_\phi\}$ используются модели, основанные на классических методах теории графов, учитывающих вероятности отказа оборудования, а описание подсистемы $\{Sb_\alpha\}$ выполняют модели подвижности абонентов [11, 13–17].

Модели подвижности подсистемы абонентского оборудования $\{Sb_\alpha\}$ разделены на два класса, класс для описания подвижности групп абонентов и класс для описания индивидуальной подвижности абонента. Описание групповой подвижности производится моделями: «Потока» (Fluid Flow Model); «Распространения» (Diffusion Model); «Гравитационной» (Gravity Model); «Перемещения абонентов по городу» (City Area Model); и др. [11]. Описание индивидуального перемещения абонента производятся моделями «Случайной прогулки» (Random Walk Model) [11] и «Случайного перемещения» (Random way point mobility model) [13], а также моделями, представленными в работах [14–17], которые направлены на прогнозирование перемещения абонента внутри сети мобильной связи. Модели групповой подвижности применяются для прогнозирования нагрузки в различных местах зоны обслуживания сети. Перераспределение нагрузки в сети происходит из-за перемещения абонентов. Модели описания индивидуальной подвижности ориентированы на решение задач «передачи» абонента между точками доступа или базовыми станциями [11], описывают поведения абонентов во время установления «peer-to-peer» связей в ad hoc сетях [13] или прогнозируют последовательность сот по которым будет перемещаться абонент [14–17].

Направленность на решение задач, связанных исключительно с прогнозированием перемещения отдельного абонента или группы абонентов по зоне обслуживания инфокоммуникационной системы, является обобщенным ограничением применения множества проанализированных моделей. Единой методологии, которая позволяет изучать и систему инфокоммуникаций и межструктурные взаимосвязи с другими системами региона, в проанализированных моделях, не выявлено. Важность выявления и формализация межструктурных взаимосвязей инфокоммуникационной системы с другими системами региона обусловлена тем, что инфокоммуникации являются «нервной» системой ряда экономических и социальных процессов.

3. Разработка метода описания системы инфокоммуникаций для реализации алгоритмического обеспечения синтеза топологических структур

Основываясь на результатах исследований моделей топологических структур инфокоммуникационных систем, предлагается метод, основанный на использовании понятия виртуального сетевого узла. Метод заключается в разделении земной поверхности на участки s_i определенного размера. Участок имеет форму прямоугольника либо правильного многоугольника. В рамках описываемых исследований выбран участок прямоугольной формы. Совокупность прямоугольников $\{s_i\}$, где i — это номер элемента, образует сеть. В каждом прямоугольнике, в течение дискретного момента времени $\Delta T = const$, находится определенное множество сетевых узлов $\{v_j\}$, j — это номер:

$$\{v_j\} \subset s_i . \tag{6}$$

В результате суммы ΔT образует условный период повторения состояния виртуального узла — T:

$$T = const : \sum_n \Delta T_n = T : n \neq \infty \wedge n \in \mathrm{N}, \tag{7}$$

где n — порядковый номер ΔT, $n \in \mathrm{N}$, означает, что n — относится к множеству натуральных чисел. На момент проведения исследований полагается, что $T = 88400$ секунд (1 сутки). Считается что, $\forall \Delta T$ между каждой парой узлов множества $\{v_j\}$, существует множество каналов $\{e_{j\,LINE}\}$ передачи данных. Топология сети внутри s_i стремится к связанности по принципу каждый с каждым. С учетом выражения (6), объединение множество $\{v_j\}$ и $\{e_{j\,LINE}\}$ образует виртуальный узел V_{Ei}:

$$V_{Ei} = [\{v_j\} \cup \{e_{j\,LINE}\} \subset s_i]. \tag{8}$$

Совокупность V_{Ei}, для всей поверхности и $\forall \Delta T$, образует множество виртуальных узлов $\{V_{Ei}\}$. Аналогично обычному сетевому узлу, одной из основных характеристик виртуального узла является пропускная способность — u_i. Изменение пропускной способности узла V_{Ei}, в течение T — дискретная функция U_i. Значение функции U_i зависит от состава узла V_{Ei} в моменты времени ΔT:

$$U_i = f(V_{Ei}, \Delta T). \tag{9}$$

Обобщая выражения (8) и (9) можно сделать вывод, что зона обслуживания сети $\forall \Delta T$ представляет поверхность S, покрытую множеством виртуальных узлов $\{V_{Ei}\}$.

Каждому элементу множества $\{V_{Ei}\}$, соответствует элемент множества $\{u_i\}$ — пропускные способности виртуальных узлов:

$$\forall \Delta T \exists \ [\{V_{Ei}\} \Leftrightarrow \{u_i\}] \subset S. \qquad (10)$$

$\forall \Delta T$ соотношение (10), на примере девяти виртуальных узлов показано на рис. 1.

Передача информации между виртуальными узлами сети осуществляется взаимодействием интерфейсов отдельных элементов V_{Ei} с интерфейсами элементов соседних V_{Ei+1}. Прогноз значений зависимости (9) реализуется сбором и обработкой текущей и ретроспективной информации о статистике посещения физическими узлами территории s_i. Точность прогноза корректируется накоплением статистики посещения территории виртуального узла физическими узлами. Прогноз может учитывать информацию о дополнительных воздействиях внешней социальной, экономической и экологической среды, которая влияет на посещаемость территории виртуального узла.

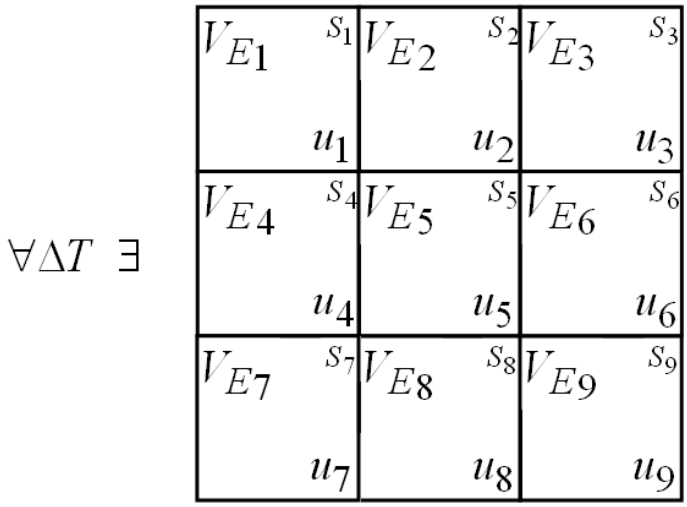


Рис. 1. Описание совокупности виртуальных узлов $\forall \Delta T$

Формирование и обработка базы статистических данных (БД) происходит поэтапно:

1. Предварительный мониторинг сети для накопления статистики посещения территории виртуальных узлов физическими (сетевыми) узлами разной степени подвижности. В результате выполнения операции накапливается база данных состояния виртуального узла. Для достоверного прогноза объем выборки базы данных υ_{Db} должен быть не ниже определенного значения $\upsilon_{Db\min}$, на формирование которой затрачивается определенный интервал времени τ значение которого должно быть не менее $\tau_{Db\min}$ (интервал времени необходимого для формирования $\upsilon_{Db\min}$). Длительность $\tau_{Db\min}$ зависит от стабильности повторяемости количества узлов в зоне развертывания сети:
$$\upsilon_{Db} \geq \upsilon_{Db\min} : \tau \geq \tau_{Db\min}. \qquad (11)$$
2. Определение вероятности, того, что количество сетевых узлов $\{v_j\}$ будет не менее ζ в момент ΔT_n, который наступает с заданной периодичностью T.
3. С учетом количества узлов V_{Ei}, выражения (9) составляются карты (топологии) пропускных способностей различных виртуальных узлов для всех ΔT.

4. На основе полученной топологии формируется матрицы $[M_n]$ состояния зоны обслуживания $\forall \Delta T$.
5. При помощи матриц $[M_n]$ определяются параметры для управления сетью: маршруты передачи трафика между абонентскими терминалами без участия сети оператора, собирается информация о состоянии зоны покрытия систем мобильной связи и др.
6. Корректировка БД о пропускных способностях V_{E_i} и оптимизация длительности T по критерию повторяемости пропускной способности виртуальных узлов.

Фактически условие (11) означает ограничение используемого формального метода при описании топологии сетей инфокоммункиаций и то, что для работы модели необходим первоначальный анализ повторяемости количества физических узлов на территории виртуального узла за некоторый промежуток времени. Как следствие, подобная модель, в представленном виде, не подходит для описания классических ad hoc сетей, развертывание которых происходит спонтанно, а статистика перемещения узлов, по области развертывания, не сформирована.

Проведение экспериментов по мониторингу наличия абонентов показало, что в отдельные ΔT число элементов множества $\{v_j\}$ (физических узлов) равно нулю. Следовательно, пропускная способность виртуального узла V_{E_i}, становилась равной нулю. Для выполнения наличия минимальной пропускной способности:

$$\forall \Delta T : U_{i\min} \neq 0, \tag{12}$$

необходимо, чтобы в виртуальном узле V_{E_i} находился хотя бы один статический узел ω_{vs}, который может выполнять функцию контроллера. Контроллер, в соответствии с номером ΔT производит рассылку информации о настройках узла V_{E_i}, вновь прибывшим физическим узлам. Совокупность ω_{vs} образует множество узлов-контроллеров $\{\omega_{vs}\}$. Узлы контроллеры связаны отдельными каналами связи e_ω. При этом $e_\omega \subset \{E_\omega\}$, где $\{E_\omega\}$ — множество каналов связи между узлами контроллерами. Множество узлов-контроллеров $\{\omega_{vs}\}$, соединенных каналами связи $\{e_\omega\}$ образуют виртуальный узел Ω_E более высокого уровня иерархии и т.д.:

$$\{\omega_{vs}\} \cup \{e_\omega\} \subset \Omega_E. \tag{13}$$

В результате виртуальная топология сети получает свойства иерархичности и самоподобия. В зависимости от технических особенностей физических узлов, входящих в состав виртуального узла, роль контроллеров может выполнять оборудование: аналогичное оборудованию подсистем доступа беспроводных сетей передачи данных, систем мобильной связи 2-го и последующих поколений, сетей спутниковой связи. Расположение подобного оборудования реализуется на вышках (зданиях), аэроплатформах, искусственных спутниках земли. Обобщенно процесс моделирования топологии сети реализуется при помощи представления в виде многослойного графа, состоящего из множества виртуальных узлов различных уровней иерархии. Для случая когда, виртуальные узлы не обладают общей границей, передача информации между ними осуществляется по маршруту, сформированному из промежуточных виртуальных узлов.

Таким образом, основа метода формирования топологии сети заключается в получении системы связи с единой логической структурой. В состав структуры входят физические узлы с различной подвижностью и пропускной способностью. При помощи

функционалов (6–10) прогнозируется пропускная способность сети из виртуальных узлов. Виртуальные узлы образуются за счет нахождения на определенной территории физических узлов. Необходимым условием применимости предложенного метода является наличие накопленной базы данных по статистке перемещения узлов в зоне обслуживания сети, что описывается выражением (11). Условием достаточности работоспособности, является выражение (13), которое требует наличия узлов более высокого уровня иерархии, которые выполняют функцию контроллера и при необходимости обеспечивают минимальную пропускную способность узлов более низкого ранга.

В целом область использования разработанной модели распространяется на различные инфокоммуникационные системы сети сотовой связи, проводные и беспроводные сети передачи данных, распределенные вычислительные комплексы и системы. Подобная широкая применимость метода достигается за счет представления топологии сети не в виде отдельных сегментированных структур, а в виде синтезированной виртуальной топологии. В подобной виртуальной топологии пространственные изменения положения элементов в реальной сети представляются в виде изменения пропускной способности неподвижных виртуальных узлов.

Обобщенно отличие предложенного метода описания системы инфокоммуникаций от методов и моделей, представленных в работах [6, 13–17], а также моделях, описывающих индивидуальную подвижность абонентов в работе [11], заключается в отсутствии необходимости мониторинга и непрерывного прогнозирования перемещения каждого узла (абонента). Процесс мониторинга и прогнозирования индивидуального перемещения абонентов замещается накоплением, обработкой и систематизаций ретроспективной информации о посещаемости определенных участков зоны покрытия сети (виртуальных узлов), что сокращает объем вычислительных ресурсов. Отличие от моделей, описывающих групповую подвижность абонентов, представленных в [11], заключается в представлении зоны обслуживания сети в виде покрытия из конечного множества элементов определенной формы (виртуальных узлов) меньшего размера, в сравнении с размером улиц или районов. Подобное представление позволяет более точно определять и прогнозировать состояние сети в заданной зоне обслуживания.

Учитывая, что начальным элементом иерархии является мобильные устройства пользователей, то сбор и обработка информации, о пользователе позволяет дополнительно показывать и обобщать социальные и экономические интересы абонентов на заданной территории. Таким образом, предлагаемый метод моделирования инфокоммуникационной сети закладывает принципы для создания единой методологии, которая позволяет изучать не только систему инфокоммуникаций в отдельности, но и межструктурные взаимосвязи в системах региона, на основе информации собираемой о пользователях телекоммуникационной сети.

4. Практическая реализация алгоритмического обеспечения для формирования виртуальной топологической структуры

Для реализации метода формирования топологической структуры в качестве источника сведений о месте расположения физических узлов использовался ресурс FlightRadar24 (URL: http://www.flightradar24.com). FlightRadar24 позволяет наблюдать перемещение воздушных судов. В число получаемых с ресурса сведений входят: коорди-

наты воздушного судна, скорость перемещения, номер рейса, начальный и конечный пункты назначения, тип и бортовой номер. В качестве допущения принято, что участие в виртуальном узле не зависит от высоты полета и типа объекта. Первым этапом является задание при помощи градусной сетки границ виртуального узла, $x_a \wedge x_b$ и $y_a \wedge y_b$, аналогично примеру, показанному на рис. 2.

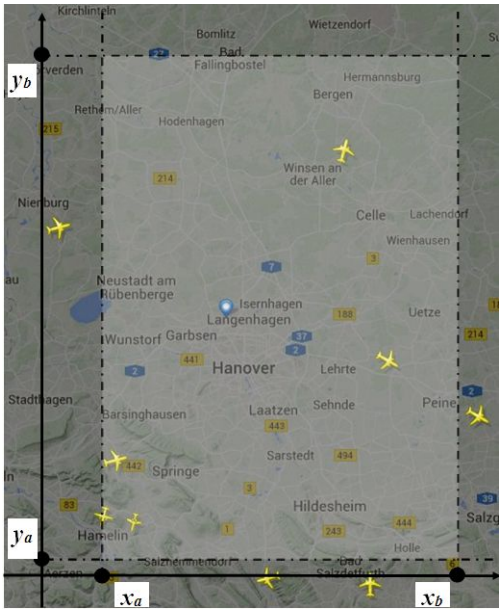


Рис. 2. Пример выделения виртуального узла на территории поверхности Земли

После определения границ на основе текущей информации о физических узлах формируется БД состояния виртуального узла:

$$Db = \{\xi\} : \xi_i(t_i; x_i; y_i, \vartheta_i) \wedge \xi \in (0, \Xi] : \Xi \neq \infty \wedge \Xi \in \mathrm{N}, \tag{14}$$

где ξ — параметр, описывающий свойства физического узла, $t_i \in [t_{start}, t_{stop}]$ — момент времени приема данных о состоянии физического узла, t_{start} и t_{stop} — время начала и окончания наблюдения, $x_i \wedge y_i : x_i \in [x_a, x_b] \wedge y_i \in [y_a, y_b]$ — координаты нахождения физического сетевого узла, ϑ_i — уникальный идентификатор физического сетевого узла (например, бортовой номер самолета).

При помощи соотношения (7) задается интервал дискретизации:

$$\Delta T = const : \sum_n \Delta T = T : n \neq \infty \wedge n \in \mathrm{N},$$

где $T = 86400\,\mathrm{c}$ (1 сутки), а $\Delta T = 900\,\mathrm{c}$ (15 мин).

При этом должно выполняться условие пропорциональности:

$$\sum_m T_m = T_{obs} : T_{obs} = t_{stop} - t_{start} \wedge m \neq \infty \wedge m \in \mathrm{N}, \tag{15}$$

где m — номер условного периода повторения состояния виртуального узла в процессе всего наблюдения T_{obs} — время наблюдения (obs. от анг. observation – наблюдение). Затем производится выделение непересекающихся множеств БД по каждому T из общего множества БД за весь период наблюдения. Например, БД за T_1 (первые сутки наблюдения) имеет вид:

$$Db_{T_1} = \{Db\}_{\Delta T_1} \cup \{Db\}_{\Delta T_2} \cup .... \cup \{Db\}_{\Delta T_n}.$$

За весь период наблюдения база данных примет вид:

$$\left.\begin{array}{l} Db_{T_1} = \{Db\}_{\Delta T_1} \cup \{Db\}_{\Delta T_2} \cup .... \cup \{Db\}_{\Delta T_n} \\ Db_{T_2} = \{Db\}_{\Delta T_1} \cup \{Db\}_{\Delta T_2} \cup .... \cup \{Db\}_{\Delta T_n} \\ ..... \\ Db_{T_m} = \{Db\}_{\Delta T_1} \cup \{Db\}_{\Delta T_2} \cup .... \cup \{Db\}_{\Delta T_n} \end{array}\right\} \rightarrow [t_{stop}, t_{start}] \quad (16)$$

В завершении при помощи индикаторной функции рассчитывается вероятность, что в составе $\{Db\}_{\Delta T}$, число физических узлов η будет не менее определенного числа ζ.

$$P(\eta \geq \zeta) = \frac{1}{\mu} \sum_{i=\zeta}^{\mu} \chi(\eta_i \geq \zeta) : \chi(\eta_i \geq \zeta) = 1 \wedge \chi(\eta_i < \zeta) = 0 \quad (17)$$

Расчет производится для каждого значения $\zeta \in [1, \zeta_{\max}]$, где $\zeta_{\max}$ максимальное количество физических узлов внутри виртуального узла за период наблюдения T_{obs}. На основании формул (14–17) и с использованием соотношений (6–8) составлен алгоритм и программа (скриптовый файл) для получения и обработки данных с ресурса FlightRadar24. Использование разработанной программы позволило спрогнозировать количество физических узлов в квадрате, показанном на рис. 2 в различные моменты времени ΔT результаты приведены в таблице.

В таблице показана вероятность появления в виртуальном узле физических объектов в виде самолетов в течение 20 суток наблюдения. Аналогичная ситуация наблюдалась в соседних виртуальных узлах. На основании эксперимента можно утверждать, что обобщение данных о состоянии соседних виртуальных узлов позволяет построить общую карту покрытия территории, аналогично описанной на рис. 1 и сформировать матрицу вероятностей наличия заданных пропускных способностей между парами виртуальных узлов.

Таблица

Вероятность нахождения не менее определенного количества физических узлов в заданном квадрате

№ $\Delta T = 900$ с	Время	$P(\eta \geq 1)$	$P(\eta \geq 2)$	$P(\eta \geq 3)$	$P(\eta \geq 4)$
4	1:00	0,952	0,905	0,857	0,81
5	1:15	0,952	0,905	0,857	0,81
6	1:30	0,952	0,905	0,857	0,81
7	1:45	0,952	0,905	0,857	0,857
8	2:00	0,952	0,905	0,857	0,857
9	2:15	0,952	0,905	0,857	0,857
10	2:30	0,952	0,905	0,8577	0,857

Особенность алгоритма — отсутствие привязки к особенностям траектории перемещения объектов, важной составляющей является факт наличия физического объекта на территории виртуального узла.

Заключение

В работе показано, что одной из проблем моделирования систем телекоммуникаций, является неопределенность отношения элемента к определенной подсистеме сети передачи информации. Неопределенность возникла из-за особенностей работы современных

систем телекоммуникаций. В современных системах один элемент может выступать как абонентское оборудование, подсистема доступа, подсистема управления и подсистема транспортной сети. В результате затрудняется внедрение информационных технологий, основанных на использовании самоорганизующихся распределенных систем передачи и обработки информации, предназначенных для увеличения доступности систем связи общего пользования и автоматизации управления транспортными средствами.

Для устранения описанной проблемы разработан метод описания структуры системы инфокоммуникаций. Метод заключается в представлении инфокоммуникационной системы как единой логической структуры. Основу структуры составляют виртуальные узлы — части земной поверхности, на которой находятся физические узлы. Физические узлы — это оборудование абонентов и провайдеров. Совокупность виртуальных узлов образует многослойную иерархическую виртуальную сеть. Пропускная способность полученной сети изменяется по вероятностным законам, которые определяются статистикой посещения виртуального узла физическими узлами. Область использования полученных статистических данных различна. Применительно к системам связи — это повышение эффективности управления структурой сети за счет прогнозирования альтернативных маршрутов передачи информации между абонентами, решения задач балансировки трафиковой нагрузки, перераспределения канального ресурса. Результаты анализа зоны обслуживания систем инфокоммуникаций можно использовать для исследования взаимодействий социальных и экономических систем региона.

Для демонстрации реализуемости разработанных положений проведен эксперимент по прогнозированию количества физических узлов в виртуальном узле. Источником информации выбран интернет ресурс FlightRadar24 — Live AIR Traffic. По полученным данным рассчитана вероятность появления в виртуальном узле физических объектов. Обобщение данных о состоянии территории наблюдения позволяет построить карту посещаемости территории виртуальных узлов физическими узлами, а затем рассчитать матрицы вероятностей пропускных способностей между парами виртуальных узлов.

Результаты исследований открывают возможности повышения эффективности использования современных телекоммуникаций, за счет синтеза логической структуры из различных объектов передачи и обработки информации.

Литература

1. Shih-Lin, Wu. Wireless ad hoc networking, Personal-Area, Local-Area, and the Sensory-Area Networks / W. Shih-Lin, T. Yu-Chee — Auerbach Publications Taylor & Francis Group New York 2007. — 660 p. DOI: 10.1201/9781420013825.
2. Moustafa, H. Vehicular Networks Techniques, Standards, and Applications / H. Moustafa Y. Zhang — Auerbach Publications Taylor & Francis Group 6000 Broken Sound Parkway NW, Suite 300 2009. — 441 p.
3. Zhang, Y. Wireless mesh networking Architectures, Protocols and Standards / Y. Zhang, J. Luo, H. Hu. — Taylor & Francis Group New York 2007. — 610 p. DOI: 10.1201/9781420013542.
4. Shorey, R. Mobile, wirless, and sensor networks technology, applications, and future directions / R. Shorey, A. Ananda, C. Mun, T. Wei — John Wiley & Sons, Inc., Hoboken, New Jersey. 2006. — 452 p. DOI: 10.1002/0471755591.

5. Сорокин А.А. Разработка программного комплекса для исследования телекоммуникационных систем с динамической топологией сети / А.А. Сорокин // Вестн. Астрахан. гос. техн. ун-та. Сер.: Управление, вычислительная техника и информатика. — 2011. — № 2. — С. 137–142
6. Hekmat, R. Ad-hoc networks: Fundamental properties and network topologies; Technology / R. Hekmat — The Netherlands and Rhyzen Information and Consulting Services, Zoetermeer, the Netherlands. 2006. — 546 p.
7. Мочалов, В.П. Модель системы управления услугами TMN / В.П. Мочалов // Современные наукоемкие технологии. — 2005. — № 5. — С. 71–72.
8. Володина, Е.В. Социально-демографический таргетинг контекстной рекламы информационных ресурсов / Е.В. Володина, П.А. Ермакова // Вестник Курганского государственного университета. Сер.: Гуманитарные науки. — Курган: Изд-во Курганского гос. ун-та. — 2011. — № 22. — С. 6–9.
9. Комашинский, В.И. Системы подвижной радиосвязи с пакетной передачей информацией. Основы моделирования. / В.И. Комашинский, А.В. Максимов — М.: Горячая линия Телеком, 2007. — 176 с.
10 Дмитриев, В.Н. Многофакторный анализ социально-экономического состояния Республики Чад для создания современной инфокоммуникационной инфраструктуры / В.Н. Дмитриев, Юсуф Ахмат, А.А. Сорокин // Вестн. Астрахан. гос. техн. ун-та. Сер.: Управление, вычислительная техника и информатика. — 2015. — № 1. — С. 56–65
11. Mukherjee, A. Location Management and Routing in Mobile Wireless Networks / A. Mukherjee, S. Bandyopadhyay, D. Saha - Artech House Boston London 2003. — 250 p.
12. Бахтин, А.А. Эффективность реализации межуровневого взаимодействия для протокола быстрой маршрутизации в беспроводных Ad-hoc сетях. / А.А. Бахтин, Л.А. Попов, А.В. Смирнов // Вестник Московского авиационного института — 2009. — № 5. — Т. 16. — С. 159–165
13. Hyytiä, E. Random Waypoint Model in n-Dimensional Space / E. Hyytiä, J. Virtamo // Operations Research Letters. — 2005. — Vol. 33/6, — P. 567–571. DOI: 10.1016/j.orl.2004.11.006.
14. Das, S. Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. / S. Das, K. Sanjoy // The Computer Journal. — 1999. — Vol. 42/6, — P. 473–486. DOI: 10.1093/comjnl/42.6.473.
15. Gökhan, Y. A data mining approach for location prediction in mobile environments. / Y. Gökhan, D. Katsaros, Ö. Ulusoy, Y. Manolopoulos // Data & Knowledge Engineering. — 2005. — Vol. 54/2, — P. 121–146. DOI: 10.1016/j.datak.2004.09.004.
16. Cheng, C. Location prediction algorithms for mobile wireless systems. / C. Cheng, R. Jain, E. Berg // In Wireless internet handbook - CRC Press, Inc. — 2003. — P. 245–263. DOI: 10.1201/9780203011690.ch11.
17. Liu, T. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. / T. Liu, B. Paramvir, C. Imrich // Selected Areas in Communications, IEEE Journal. — 1998. — Vol. 16/6 — P. 922–936. DOI: 10.1109/49.709453.

Сорокин Александр Александрович, к.т.н., доцент кафедры «Связь», Астраханский государственный технический университет (Астрахань, Российская Федерация), alsorokin2@list.ru.

Резников Петр Сергеевич, аспирант, ассистент кафедры «Связь», Астраханский государственный технический университет (Астрахань, Российская Федерация), psreznikov@gmail.com.

Поступила в редакцию 9 июля 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 4, pp. 64–79

DOI: 10.14529/cmse150404

DEVELOPMENT ALGORITHMIC SUPPORT FOR THE SYNTHESIS OF TOPOLOGICAL STRUCTURES OF COMMUNICATION SYSTEMS

A.A. Sorokin, Astrakhan State Technical University (Astrakhan, Russian Federation) alsorokin2@list.ru,

P.S. Reznikov, Astrakhan State Technical University (Astrakhan, Russian Federation) psreznikov@gmail.com

The paper shows that one of the problem of telecommunication systems is indeterminacy of relation of a system's element to a particular subsystem of informational network, when the element could be used either as a user equipment or a management subsystem. The result is the difficulty of re-introduction of ad hoc networks into the structure of the mobile communication systems. To solve this problem a method of describing the structure of information communications has been developed. The method is based on the presentation of the system as a single logical structure. The base of the structure are virtual nodes. A set of virtual nodes forms a virtual network. The capacity of a virtual network varies according to the laws of probability. Probability laws are determined on the basis of statistics of virtual host attendance by physical nodes. The sphere of application of communications systems is the improvement of management efficiency due to anticipating alternative routes of data transmission and different standards of communication systems synthesis. The results can be used to study the interactions of social and economic systems of the region. To demonstrate the results the experiment on the prediction of the number of physical nodes in a given virtual host has been carried out. The probability of occurrence of physical objects in the virtual site has been calculated on the bases of the obtained data. The calculation results enable us to construct a map of the area attendance by physical nodes and to calculate traffic routes between virtual hosts.

Keywords: system, analysis, infocommunications, topology, structure, synthesis, information, management, processing, network.

References

1. Shih-Lin W., Yu-Chee T. Wireless ad hoc networking, Personal-Area, Local-Area, and the Sensory-Area Networks. Auerbach Publications Taylor & Francis Group New York, 2007. 660 p. DOI: 10.1201/9781420013825.
2. Moustafa H. Zhang Y. Vehicular Networks Techniques, Standards, and Applications. Auerbach Publications Taylor & Francis Group 6000 Broken Sound Parkway NW, Suite 300, 2009. 441 p.
3. Zhang Y., Luo J., Hu H. Wireless mesh networking Architectures, Protocols and Standards. Taylor & Francis Group New York, 2007. 610 p. DOI: 10.1201/9781420013542.
4. Shorey R., Ananda A., Mun C., Wei T. Mobile, wirless, and sensor networks technology, applications, and future directions. John Wiley & Sons, Inc., Hoboken, New Jersey, 2006. 452 p. DOI: 10.1002/0471755591.
5. Sorokin A.A. Razrabotka programmnogo kompleksa dlja issledovanija telekommunikacionnyh sistem s dinamicheskoj topologiej seti [Development of software for the study of telecommunication systems with dynamic network topology] // Vestn. Astrahan. gos. tehn. un-ta. Ser.: Upravlenie, vychislitel'naja tehnika i informatika [Bulletin of the Astrakhan State. tehn. Univ. Ser .: Management, Computer Science and Informatics]. 2011. Vol. 2. P. 137–142
6. Hekmat R. Ad-hoc networks: Fundamental properties and network topologies; Technology. The Netherlands and Rhyzen Information and Consulting Services, Zoetermeer, the Netherlands, 2006. 546 p.
7. Mochalov V.P. Model' sistemy upravlenija uslugami TMN [Model of service management system TMN] // Sovremennye naukoemkie tehnologii [Modern high technologies]. 2005. № 5. P. 71–72.
8. Volodina, E.V., Ermakova, P.A. Social'no-demograficheskij targeting kontekstnoj reklamy informacionnyh resursov [Socio-demographic targeting of contextual advertising information resources] // Vestnik Kurganskogo gosudarstvennogo universiteta. Ser.: Gumanitarnye nauki [Bulletin of the Kurgan State University. Series: Humanities]. Kurgan: Publishing house Kurgan State. University Pres. 2011. № 22. P. 6–9.
9. Komashinskij V.I., Maksimov A.V., Sistemy podvizhnoj radiosvjazi s paket-noj peredachej informaciej. Osnovy modelirovanija [Mobile radio systems with packet transmission of information. Fundamentals of modeling.]. M .: Hotline Telecom, 2007. 176 p.
10. Dmitriev V.N., Ahmat Jusuf, Sorokin A.A. Mnogofaktornyj analiz social'no-jekonomicheskogo sostojanija Respubliki Chad dlja sozdanija sovremennoj infokommunikacionnoj infrastruktury [Multivariate analysis of social and economic state of the republic of chad for formation of the modern communication infrastructure] // Vestn. Astrahan. gos. tehn. un-ta. Ser.: Upravlenie, vychislitel'naja tehnika i informatika [Bulletin of the Astrakhan State. tehn. Univ. Ser .: Management, Computer Science and Informatics]. 2015. Vol. 1. P. 56–65
11. Mukherjee A, Bandyopadhyay S., Saha D. Location Management and Routing in Mobile Wireless Networks - Artech House Boston London, 2003. 250 p.
12. Bahtin A.A., Popov L.A., Smirnov A.V. Jeffektivnost' realizacii mezhurovnevogo vzaimodejstvija dlja protokola bystroj marshrutizacii v besprovodnyh Ad-hoc setjah [The effectiveness of the implementation of the interlayer interaction for fast routing protocol

for wireless Ad-hoc networks] // Vestnik Moskovskogo aviacionnogo institute [Bulletin of the Moscow Aviation Institute]. 2009. Vol. 5, ie. 16 P. 159–165

13. Hyytiä E., Virtamo J. Random Waypoint Model in n-Dimensional Space // Operations Research Letters, 2005. Vol. 33/6. P. 567–571. DOI: 10.1016/j.orl.2004.11.006.
14. Das S., Sanjoy K. Adaptive location prediction strategies based on a hierarchical network model in a cellular mobile environment. // The Computer Journal, 1999. Vol. 42/6. P. 473–486. DOI: 10.1093/comjnl/42.6.473.
15. Gökhan Y., Katsaros D., Ulusoy Ö. Manolopoulos Y. A data mining approach for location prediction in mobile environments. // Data & Knowledge Engineering, 2005. Vol. 54/2. P. 121–146. DOI: 10.1016/j.datak.2004.09.004.
16. Cheng C., Jain R., Berg E. Location prediction algorithms for mobile wireless systems. // In Wireless internet handbook, CRC Press, Inc., 2003. P. 245–263. DOI: 10.1201/9780203011690.ch11.
17. Liu T., Paramvir B., Imrich C. Mobility modeling, location tracking, and trajectory prediction in wireless ATM networks. // Selected Areas in Communications, IEEE Journal, 1998. Vol. 16/6. P. 922–936. DOI: 10.1109/49.709453.

Received July 9, 2015.

УДК 004.657 DOI: 10.14529/cmse150405

ПАРАЛЛЕЛЬНАЯ ДЕКОМПОЗИЦИЯ РЕЛЯЦИОННЫХ ОПЕРАЦИЙ НА ОСНОВЕ РАСПРЕДЕЛЕННЫХ КОЛОНОЧНЫХ ИНДЕКСОВ

Е.В. Иванова, Л.Б. Соколинский

Данная статья является продолжением и развитием более ранней работы авторов, в которой была рассмотрена декомпозиция операций пересечения и соединения колоночных индексов на основе доменно-интервальной фрагментации. Такая декомпозиция позволяет организовать параллельное выполнение реляционных операций над распределенными колоночными индексами без массовых обменов данными между процессорными узлами. В настоящей статье рассматривается декомпозиция операций проекции, выбора, удаления дубликатов и объединения. Кроме этого, вводится новый вид колоночных индексов, названных колоночными хеш-индексами. Колоночный хеш-индекс способен индексировать сразу несколько атрибутов отношения. Для распределенных колоночных хеш-индексов рассматривается декомпозиция операций пересечения, объединения и естественного соединения.

Ключевые слова: распределенные колоночные индексы, доменно-интервальная фрагментация, колоночные хеш-индексы, декомпозиция реляционных операций.

Введение

В последнее время большой интерес у исследователей вызывают системы баз данных с хранением баз данных по столбцам. Такое представление данных называется *колоночным* [1, 2]. Колоночное представление в отличие от традиционного строкового представления оказывается намного более эффективным при выполнении запросов класса OLAP [3]. Это объясняется тем, что при выполнении запросов колоночная СУБД считывает с диска только те атрибуты, которые необходимы для выполнения запроса, что сокращает объем операций ввода-вывода и, как следствие, уменьшает время выполнения запроса. Недостатком колоночного представления является низкая эффективность при выполнении строково-ориентированных операций, таких, например, как добавление или удаление кортежей. Вследствие этого колоночные СУБД могут проигрывать по производительности строковым при выполнении запросов класса OLTP. Одним из главных преимуществ строчных хранилищ является наличие в строковых СУБД мощных процедур оптимизации запросов, разработанных на базе реляционной модели. Строковые СУБД также имеют большое преимущество в скорости обработки запросов класса OLTP. В соответствии с этим в исследовательском сообществе баз данных были предприняты интенсивные усилия по интеграции преимуществ столбцовой модели хранения данных в строковые СУБД. Анализ рассмотренных решений показывает [3], что нельзя получить выгоду от хранения данных по столбцам, воспользовавшись системой баз данных со строковым хранением с вертикально разделенной схемой, либо проиндексировав все столбцы, чтобы обеспечить к ним независимый доступ.

Авторами в работах [4–10] был предложен новый подход к интеграции преимуществ строчных и колоночных хранилищ, суть которого заключается во введении распреде-

ленных колоночных индексов, хранимых в оперативной памяти кластерной вычислительной системы. Эти колоночные индексы обрабатываются с помощью программной системы, получившей название «колоночный сопроцессор КСОП». Назначение КСОП — вычислять таблицы предварительных вычислений для ресурсоемких реляционных операций по запросу СУБД. Общая схема взаимодействия СУБД и КСОП изображена на рис. 1.

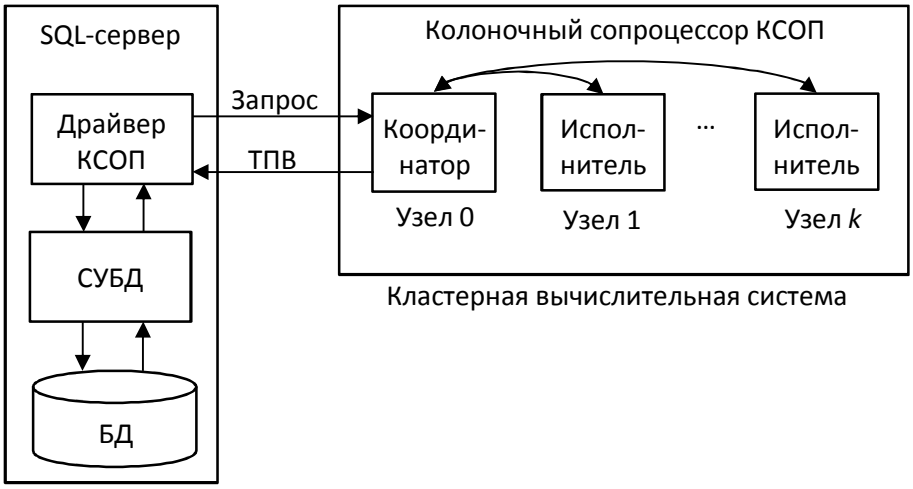


Рис. 1. Взаимодействие SQL-сервера с колоночным сопроцессором КСОП

КСОП включает в себя программу «Координатор», запускаемую на узле вычислительного кластера с номером 0, и программу «Исполнитель», запускаемую на всех остальных узлах, выделенных для работы КСОП. На SQL-сервере устанавливается специальная программа «Драйвер КСОП», обеспечивающая взаимодействие с координатором КСОП по протоколу TCP/IP.

Теоретические основы колоночного сопроцессора были изложены авторами в работе [4], где были описаны колоночные индексы и доменно-интервальная фрагментация. Суть математической модели заключается в следующем. Пусть задано отношение $R(A,B,\ldots), T(R)=n$. Атрибут A в отношении R играет роль суррогатного ключа. Пусть на множестве $\mathfrak{D}_B$, являющемся доменом атрибута B, задано отношение линейного порядка. *Колоночным индексом* $I_{R.B}$ атрибута B отношения R будем называть упорядоченное отношение $I_{R.B}(A,B)$, удовлетворяющее следующим свойствам:

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

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

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

С содержательной точки зрения колоночный индекс $I_{R.B}$ представляет собой таблицу из двух колонок с именами A и B. Количество строк в колоночном индексе совпадает с количеством строк в индексируемой таблице. Колонка B индекса $I_{R.B}$ включает в себя все значения колонки B таблицы R, отсортированных в порядке возрастания. Каждая строка x индекса $I_{R.B}$ содержит в колонке A суррогатный ключ (адрес) строки r в таблице R, имеющей такое же значение в колонке B, что и строка x. На рис. 2 представлены примеры двух различных колоночных индексов для одного и того же отношения.

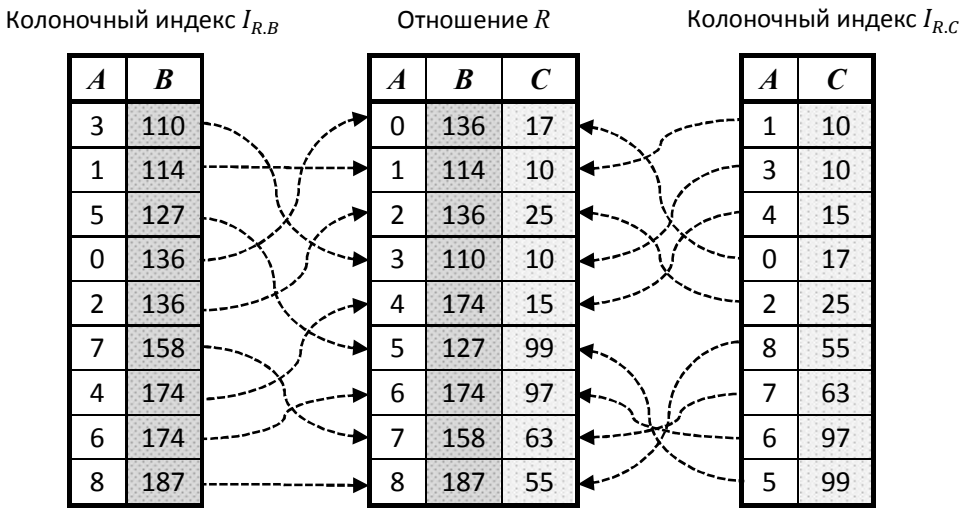


Рис. 2. Колоночные индексы

Для фрагментации колоночных индексов по узлам многопроцессорной системы авторами был предложен метод доменно-интервальной фрагментации [4], суть которого заключается в следующем. Пусть на множестве значений домена $\mathfrak{D}_B$ задано отношение линейного порядка. Разобьем множество $\mathfrak{D}_B$ на $k>0$ непересекающихся интервалов:

$$\left.\begin{array}{l} V_0=[v_0;v_1),V_1=[v_1;v_2),\ldots,V_{k-1}=[v_{k-1};v_k); \\ v_0<v_1<\ldots<v_k; \\ \mathfrak{D}_B=\bigcup\limits_{i=0}^{k-1}V_i. \end{array}\right\} \quad (4)$$

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

$$\forall i\in\{0,\ldots,k-1\}\left(\forall b\in\mathfrak{D}_B\left(\varphi_{\mathfrak{D}_B}(b)=i\Leftrightarrow b\in V_i\right)\right). \quad (5)$$

Пусть задан колоночный индекс $I_{R.B}$ для отношения $R(A,B,\ldots)$ с атрибутом B над доменом $\mathfrak{D}_B$ и доменная функция фрагментации $\varphi_{\mathfrak{D}_B}$. Функция

$$\varphi_{I_{R.B}}:I_{R.B}\to\{0,\ldots,k-1\}, \quad (6)$$

определенная по правилу

$$\forall x\in I_{R.B}\left(\varphi_{I_{R.B}}(x)=\varphi_{\mathfrak{D}_B}(x.B)\right), \quad (7)$$

называется *доменно-интервальной функцией фрагментации* для индекса $I_{R.B}$. Другими словами, функция фрагментации $\varphi_{I_{R.B}}$ сопоставляет каждому кортежу x из $I_{R.B}$ номер доменного интервала, которому принадлежит значение $x.B$.

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

$$I^i_{R.B}=\left\{x\mid x\in I_{R.B};\ \varphi_{I_{R.B}}(x)=i\right\}. \quad (8)$$

Это означает, что в i-тый фрагмент попадают кортежи, у которых значение атрибута B принадлежит i-тому доменному интервалу. Фрагментация, построенная таким образом, называется *доменно-интервальной*. Количество фрагментов k называется *степенью*

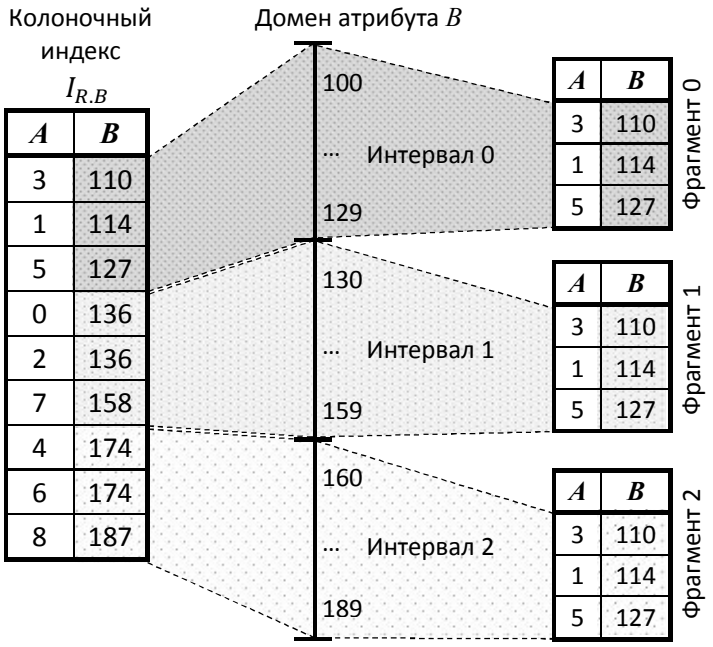


Рис. 3. Фрагментация колоночного индекса

фрагментации. На рис. 3 схематично изображена доменно-интервальная фрагментация колоночного индекса, имеющая степень $k=3$.

Пусть для отношения $R(A,B,C,\ldots)$ заданы колоночные индексы $I_{R.B}$ и $I_{R.C}$. *Транзитивной фрагментацией* индекса $I_{R.C}$ относительно индекса $I_{R.B}$ называется фрагментация, задаваемая функцией $\ddot{\varphi}_{I_{R.C}} : I_{R.C} \to \{0,\ldots,k-1\}$, удовлетворяющей условию $\forall x \in I_{R.C}$:

$$\ddot{\varphi}_{I_{R.C}}(x) = \varphi_{I_{R.B}}\left(\sigma_{A=x.A}\left(I_{R.B}\right)\right). \tag{9}$$

Транзитивная фрагментация позволяет разместить на одном и том же узле элементы колоночных индексов, соответствующие одному кортежу индексируемого отношения.

В работе в работе [4] авторами была выполнена декомпозиция операций пересечения и соединения, в [8] — операции группировки. В настоящей статье рассматривается декомпозиция операций проекции, выбора, удаления дубликатов и объединения. Кроме этого, вводится новый вид колоночных индексов, названных колоночными хеш-индексами. Колоночный хеш-индекс способен индексировать сразу несколько атрибутов отношения. Для распределенных колоночных хеш-индексов рассматривается декомпозиция операций пересечения, объединения и естественного соединения.

Статья имеет следующую структуру. В разделе 1 рассматривается декомпозиция операций проекции, выбора, удаления дубликатов и объединения с использованием фрагментированных колоночных индексов. В разделе 2 вводится понятие колоночного хеш-индекса и декомпозиция операций пересечения, объединения и естественного соединения с использованием распределенных хеш-индексов. В заключении суммируются полученные результаты, делаются итоговые выводы.

1. Декомпозиция реляционных операций с использованием фрагментированных колоночных индексов

В данном разделе рассматривается декомпозиция операций проекции, выбора, удаления дубликатов и объединения с использованием фрагментированных колоночных индексов.

1.1. Декомпозиция проекции

В этом разделе рассматривается декомпозиция операции проекции вида $\pi_{B,C_1,\ldots,C_u}(R)$. Алгоритм выполнения проекции в строчных СУБД не содержит ресурсоемких вычислений, однако он требует чтения с диска всего отношения. В отличие от этого, предлагаемый ниже алгоритм использует только те столбцы (колоночные индексы) отношения, которые вовлекаются в проекцию. Кроме этого, он вообще не предполагает чтений с диска, так как колоночные индексы хранятся в оперативной памяти.

Пусть задано отношение $R(A,B,C_1,\ldots,C_u,\ldots)$ с суррогатным ключом A. Пусть имеется колоночный индекс $I_{R.B}(A,B)$. Пусть также имеются колоночные индексы $I_{R.C_1}(A,C_1),\ldots,I_{R.C_u}(A,C_u)$. Пусть для индекса $I_{R.B}$ задана доменно-интервальная фрагментация степени k:

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

Пусть для индексов $I_{R.C_1},\ldots,I_{R.C_u}$ задана транзитивная относительно $I_{R.B}$ фрагментация:

$$\forall j\in\{1,\ldots,u\}\left(I_{R.C_j}=\bigcup_{i=0}^{k-1} I_{R.C_j}^i\right). \tag{11}$$

Положим

$$P_i=I_{R.B}^i\bowtie I_{R.C_1}^i\bowtie\ldots\bowtie I_{R.C_u}^i \tag{12}$$

для всех $i=0,\ldots,k-1$. Определим

$$P=\bigcup_{i=0}^{k-1} P_i\ . \tag{13}$$

Построим отношение $Q(B,C_1,\ldots,C_u)$ следующим образом:

$$Q=\pi_{B,C_1,\ldots,C_u}(P). \tag{14}$$

Теорема 1. $Q=\pi_{B,C_1,\ldots,C_u}(R)$.

Доказательство. Сначала докажем, что

$$Q\subset\pi_{B,C_1,\ldots,C_u}(R). \tag{15}$$

Пусть $(b,c_1,\ldots,c_u)\in Q$. В силу (14) и (13) существуют a и i такие, что $(a,b,c_1,\ldots,c_u)\in P_i$. В силу (12) имеем:

$$(a,b)\in I_{R.B};$$
$$(a,c_1)\in I_{R.C_1};$$
$$\ldots\ldots\ldots$$
$$(a,c_u)\in I_{R.C_u}.$$

По определению колоночного индекса (свойства (1) и (3)) отсюда следует, что существует $r \in R$ такой, что $r.A = a; r.B = b; r.C_1 = c_1; \ldots; r.C_u = c_u$, откуда следует $(b, c_1, \ldots, c_u) \in \pi_{B,C_1,\ldots,C_u}(R)$, то есть (15) имеет место.

Теперь покажем, что

$$Q \supset \pi_{B,C_1,\ldots,C_u}(R). \tag{16}$$

Пусть $(b, c_1, \ldots, c_u) \in \pi_{B,C_1,\ldots,C_u}(R)$. Тогда существует $r \in R$ такой, что $r.B = b;\ r.C_1 = c_1;\ \ldots;\ r.C_u = c_u$. Положим $a = r.A$. По определению колоночного индекса (свойства (1) и (3)) отсюда следует что

$$\begin{gathered} (a,b) \in I_{R.B}; \\ (a,c_1) \in I_{R.C_1}; \\ \ldots\ldots\ldots \\ (a,c_u) \in I_{R.C_u}. \end{gathered} \tag{17}$$

В силу свойств доменно-интервальной фрагментации существует i такой, что $(a,b) \in I^i_{R.B}$. По определению транзитивной относительно $I_{R.B}$ фрагментации отсюда, с учетом (17), следует

$$\begin{gathered} (a,c_1) \in I^i_{R.C_1}; \\ \ldots\ldots\ldots \\ (a,c_u) \in I^i_{R.C_u}. \end{gathered}$$

Тогда, в силу (12), $(a, b, c_1, \ldots, c_u) \in P_i$. Учитывая (13), получаем $(a, b, c_1, \ldots, c_u) \in P$, то есть $(b, c_1, \ldots, c_u) \in \pi_{B,C_1,\ldots,C_u}(P)$. Принимая во внимание (14), имеем $(b, c_1, \ldots, c_u) \in Q$, что означает, что (16) имеет место. *Теорема доказана.*

1.2. Декомпозиция операции выбора

В данном разделе рассматривается декомпозиция операции выбора вида $\sigma_\theta(R)$, где θ — некоторое условие, накладываемое на атрибуты отношения R.

Пусть имеется отношение $R(A, B, C_1, \ldots, C_u, D_1, \ldots, D_w)$ с суррогатным ключом A. Рассмотрим операцию выбора вида $\sigma_{\theta(B,C_1,\ldots,C_u)}(R)$, где $\theta(B, C_1, \ldots, C_u)$ — некоторое условие, зависящее от значений атрибутов $B, C_1, \ldots, C_u$ отношения R. Пусть имеется колоночный индекс $I_{R.B}$. Пусть также имеются колоночные индексы:

$$I_{R.C_1}, \ldots, I_{R.C_u}.$$

Пусть для индекса $I_{R.B}$ задана доменно-интервальная фрагментация степени k:

$$I_{R.B} = \bigcup_{i=0}^{k-1} I^i_{R.B}. \tag{18}$$

Пусть для индексов $I_{R.C_1}, \ldots, I_{R.C_u}$ и $I_{R.D_1}, \ldots, I_{R.D_w}$ задана транзитивная относительно $I_{R.B}$ фрагментация:

$$\forall j \in \{1, \ldots, u\} \left(I_{R.C_j} = \bigcup_{i=0}^{k-1} I^i_{R.C_j} \right). \tag{19}$$

Положим

$$P_i = \pi_A \left(\sigma_{\theta(B,C_1,\ldots,C_u)} \left(I^i_{R.B} \bowtie I^i_{R.C_1} \bowtie \ldots \bowtie I^i_{R.C_u} \right) \right) \tag{20}$$

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

$$P = \bigcup_{i=0}^{k-1} P_i \ . \tag{21}$$

Построим отношение $Q(A,B,C_1,\ldots,C_u,D_1,\ldots,D_w)$ следующим образом:

$$Q = \{\&_R(p.A) \mid p \in P\} \ . \tag{22}$$

Теорема 2. $Q = \sigma_{\theta(B,C_1,\ldots,C_u)}(R)$.

Доказательство. Сначала докажем, что

$$Q \subset \sigma_{\theta(B,C_1,\ldots,C_u)}(R) \ . \tag{23}$$

Пусть $(a,b,c_1,\ldots,c_u,d_1,\ldots,d_w) \in Q$. Тогда из (22) и (1) следует, что существует $r \in R$ такой, что

$$r.A = a; r.B = b; r.C_1 = c_1; \ldots; r.C_u = c_u; r.D_1 = d_1; \ldots; r.D_w = d_w \ . \tag{24}$$

С другой стороны, в силу (22) существует $p \in P$ такой, что $(a) \in P$. Тогда в силу (21) существует i такой, что $(a) \in P_i$. Отсюда, с учетом (20), существуют $b', c'_1, \ldots, c'_u$ такие, что

$$(a,b',c'_1,\ldots,c'_u) \in \sigma_{\theta(B,C_1,\ldots,C_u)}\left(I^i_{R.B} \bowtie I^i_{R.C_1} \bowtie \ldots \bowtie I^i_{R.C_u}\right) . \tag{25}$$

Отсюда следует, что

$$\theta(b',c'_1,\ldots,c'_u) = true \tag{26}$$

и

$$\begin{gathered} (a,b') \in I_{R.B}; \\ (a,c'_1) \in I_{R.C_1}; \\ \ldots\ldots\ldots \\ (a,c'_u) \in I_{R.C_u} . \end{gathered}$$

По определению колоночного индекса (свойства (1) и (3)) отсюда следует, что существует $r' \in R$ такой, что $r'.A = a$, $r'.B = b'$, $r'.C_1 = c'_1, \ldots, r'.C_u = c'_u$. Принимая во внимание (26), тогда имеем $\theta(r'.B,r'.C_1,\ldots,r'.C_u) = true$, то есть $r' \in \sigma_{\theta(B,C_1,\ldots,C_u)}(R)$. Поскольку $r'.A = a = r.A$ и A — суррогатный ключ в R, с учетом (24), отсюда получаем $(a,b,c_1,\ldots,c_u,d_1,\ldots,d_w) \in \sigma_{\theta(B,C_1,\ldots,C_u)}(R)$, то есть (23) имеет место.

Теперь покажем, что

$$Q \supset \sigma_{\theta(B,C_1,\ldots,C_u)}(R) \ . \tag{27}$$

Пусть $(a,b,c_1,\ldots,c_u,d_1,\ldots,d_w) \in \sigma_{\theta(B,C_1,\ldots,C_u)}(R)$. Тогда существует $r \in R$ такой, что

$$r.A = a; \ r.B = b; \ r.C_1 = c_1; \ \ldots \ ; \ r.C_u = c_u; \ r.D_1 = d_1; \ \ldots \ ; \ r.D_w = d_w \tag{28}$$

и

$$\theta(b,c_1,\ldots,c_u) = true \ . \tag{29}$$

По определению колоночного индекса (свойства (1) и (3)) из (28) следует что

$$\begin{gathered} (a,b) \in I_{R.B}; \\ (a,c_1) \in I_{R.C_1}; \\ \ldots\ldots\ldots \\ (a,c_u) \in I_{R.C_u} . \end{gathered} \tag{30}$$

В силу свойств доменно-интервальной фрагментации существует i такой, что $(a,b) \in I^i_{R.B}$. По определению транзитивной относительно $I_{R.B}$ фрагментации отсюда, с учетом (30), следует

$$(a,c_1)\in I^i_{R.C_1};$$

……

$$(a,c_u)\in I^i_{R.C_u}.$$

Тогда с учетом (29) из (20) получаем $(a)\in P_i$. Принимая во внимание (21), имеем $(a)\in P$. С учетом (22) и (1) отсюда следует, что существует $r'\in R$ такой, что

$$r'.A=a \tag{31}$$

и

$$\left(r'.A,r'.B,r'.C_1,\ldots,r'.C_u,r'.D_1,\ldots,r'.D_w\right)\in Q. \tag{32}$$

Так как A — суррогатный ключ, из (31) следует, что $r'=r$. Вместе с (32) и (28) это дает $(a,b,c_1,\ldots,c_u,d_1,\ldots,d_w)\in Q$, то есть (27) имеет место. *Теорема доказана.*

1.3. Декомпозиция операции удаления дубликатов

Пусть задано отношение $R\left(A,B,C_1,\ldots,C_u\right)$ с суррогатным ключом A. Пусть имеется колоночный индекс $I_{R.B}$. Пусть также имеются колоночные индексы $I_{R.C_1},\ldots,I_{R.C_u}$. Пусть для индекса $I_{R.B}$ задана доменно-интервальная фрагментация степени k:

$$I_{R.B}=\bigcup_{i=0}^{k-1}I^i_{R.B}. \tag{33}$$

Пусть для индексов $I_{R.C_1},\ldots,I_{R.C_u}$ задана транзитивная относительно $I_{R.B}$ фрагментация:

$$\forall j\in\{1,\ldots,u\}\left(I_{R.C_j}=\bigcup_{i=0}^{k-1}I^i_{R.C_j}\right). \tag{34}$$

Положим

$$P_i=\pi_A\left(\gamma_{\min(A)\to A,B,C_1,\ldots,C_u}\left(I^i_{R.B}\bowtie I^i_{R.C_1}\bowtie\ldots\bowtie I^i_{R.C_u}\right)\right) \tag{35}$$

для всех $i=0,\ldots,k-1$. Определим

$$P=\bigcup_{i=0}^{k-1}P_i. \tag{36}$$

Построим отношение $Q(B,C_1,\ldots,C_u)$ следующим образом:

$$Q=\left\{\left(\&_R(p.A).B,\&_R(p.A).C_1,\ldots,\&_R(p.A).C_u\right)\mid p\in P\right\}. \tag{37}$$

Теорема 3. $Q=\delta\left(\pi_{B,C_1,\ldots,C_u}\left(R\right)\right)$.

Доказательство. Сначала докажем, что

$$Q\subset\delta\left(\pi_{B,C_1,\ldots,C_u}\left(R\right)\right). \tag{38}$$

Пусть $(b,c_1,\ldots,c_u)\in Q$. Тогда из (37) следует что существует a такой, что $(a)\in P$, и существует $r\in R$ такой, что

$$r.A=a;\ r.B=b;\ r.C_1=c_1;\ \ldots;\ r.C_u=c_u. \tag{39}$$

Следовательно $(b,c_1,\ldots,c_u)\in\delta\left(\pi_{B,C_1,\ldots,C_u}\left(R\right)\right)$.

Теперь покажем, что

$$Q\supset\delta\left(\pi_{B,C_1,\ldots,C_u}\left(R\right)\right). \tag{40}$$

Пусть

$$(b,c_1,\ldots,c_u)\in\delta\left(\pi_{B,C_1,\ldots,C_u}\left(R\right)\right).$$

Положим

$$R'(A,B,C_1,\ldots,C_u)=\left\{r\middle|r\in R\wedge r.B=b\wedge r.C_1=c_1\wedge\ldots\wedge r.C_u=c_u\right\}. \quad (41)$$

Вычислим кортеж

$$r'=\gamma_{\min(A)\to A,B,C_1,\ldots,C_u}\left(R'\right). \quad (42)$$

Очевидно, что

$$r'\in R \quad (43)$$

и

$$r'.B=b;\ r'.C_1=c_1;\ \ldots\ ;\ r'.C_u=c_u. \quad (44)$$

Обозначим

$$a=r'.A. \quad (45)$$

По определению колоночного индекса (свойства (1) и (3)) из (43), (44) и (45) следует что

$$(a,b)\in I_{R.B};$$
$$(a,c_1)\in I_{R.C_1};$$
$$\ldots\ldots\ldots$$
$$(a,c_u)\in I_{R.C_u}.$$

В силу свойств доменно-интервальной фрагментации существует i такой, что $(a,b)\in I^i_{R.B}$. По определению транзитивной относительно $I_{R.B}$ фрагментации отсюда следует

$$(a,c_1)\in I^i_{R.C_1};$$
$$\ldots\ldots\ldots$$
$$(a,c_u)\in I^i_{R.C_u}.$$

Тогда с учетом (42) из (35) получаем $(a)\in P_i$. Принимая во внимание (36), имеем $(a)\in P$. С учетом (37) и (1) отсюда следует, что существует $r''\in R$ такой, что

$$r''.A=a \quad (46)$$

и

$$\left(r''.B,r''.C_1,\ldots,r''.C_u\right)\in Q. \quad (47)$$

Так как A — суррогатный ключ, из (45) и (46) следует, что $r'=r''$. Вместе с (44) и (47) это дает $(b,c_1,\ldots,c_u)\in Q$, то есть (40) имеет место. *Теорема доказана.*

1.4. Декомпозиция операции объединения

В данном разделе рассматривается декомпозиция операции объединения двух отношений вида $\pi_{B_1,\ldots,B_u}(R)\cup\pi_{B_1,\ldots,B_u}(S)$. При этом предполагается, что $\pi_{B_1,\ldots,B_u}(R)$ и $\pi_{B_1,\ldots,B_u}(S)$ не содержат дубликатов. Результирующее отношение также не должно содержать дубликатов.

Пусть заданы два отношения $R\left(A,B_1,\ldots,B_u\right)$ и $S\left(A,B_1,\ldots,B_u\right)$, имеющие одинаковый набор атрибутов. Пусть имеется два набора колоночных индексов по атрибутам $B_1,\ldots,B_u$:

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

Пусть для индексов $I_{R.B_1}$ и $I_{S.B_1}$ задана доменно-интервальная фрагментация степени k:

$$I_{R.B_1}=\bigcup_{i=0}^{k-1}I^i_{R.B_1}; \quad (48)$$

$$I_{S.B_1} = \bigcup_{i=0}^{k-1} I^i_{S.B_1} \ . \tag{49}$$

Пусть для индексов $I_{R.B_2},\ldots,I_{R.B_u}$ и $I_{S.B_2},\ldots,I_{S.B_u}$ задана транзитивная фрагментация относительно $I_{R.B_1}$ и $I_{S.B_1}$ соответственно:

$$\forall j \in \{2,\ldots,u\}\left(I_{R.B_j} = \bigcup_{i=0}^{k-1} I^i_{R.B_j} \right); \tag{50}$$

$$\forall j \in \{2,\ldots,u\}\left(I_{S.B_j} = \bigcup_{i=0}^{k-1} I^i_{S.B_j} \right). \tag{51}$$

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

$$\tilde{P}^i_j = \pi_{I^i_{R.B_j}.A \to A_R,\, I^i_{S.B_j}.A \to A_S}\left(I^i_{R.B_j} \underset{(I^i_{R.B_j}.B_j = I^i_{S.B_j}.B_j)}{\bowtie} I^i_{S.B_j} \right). \tag{52}$$

Определим

$$\tilde{P}^i = \bigcap_{j=1}^{u} \tilde{P}^i_j \ . \tag{53}$$

Положим для всех $i = 0,\ldots,k-1$

$$P^i_R = \pi_A\left(I^i_{R.B_1} \right) \tag{54}$$

и

$$P^i_S = \pi_A\left(I^i_{S.B_1} \underset{(I^i_{S.B_1}.A \neq \tilde{P}^i.A_S)}{\bowtie} \tilde{P}^i \right). \tag{55}$$

Определим

$$P_R = \bigcup_{i=0}^{k-1} P^i_R \ , \tag{56}$$

$$P_S = \bigcup_{i=0}^{k-1} P^i_S \ . \tag{57}$$

Построим отношение $Q(A,B_1,\ldots,B_u)$ следующим образом:

$$Q = \left\{ \&_R\left(p.A\right) \mid p \in P_R \right\} \cup \left\{ \&_S\left(p.A\right) \mid p \in P_S \right\}. \tag{58}$$

Теорема 4. $\pi_{B_1,\ldots,B_u}(Q) = \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S)$.

Доказательство. Сначала докажем, что

$$\pi_{B_1,\ldots,B_u}(Q) \subset \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S). \tag{59}$$

Пусть

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(Q). \tag{60}$$

Из (58) следует, что как минимум одно из следующих условий истинно:

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}\left(\left\{ \&_R\left(p.A\right) \mid p \in P_R \right\} \right); \tag{61}$$

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}\left(\left\{ \&_S\left(p.A\right) \mid p \in P_S \right\} \right). \tag{62}$$

Предположим сначала, что условие (61) истинно. Тогда существует $p \in P_R$ такой, что

$$\&_R\left(p.A\right) = r \in R$$

и

$$r = (a,b_1,\ldots,b_u),$$

где $a = p.A$. Отсюда получаем $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R)$. Следовательно, $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S)$, и (59) имеет место. Для случая, когда истинным явля-

ется условие (62), это утверждение доказывается аналогично (в рассуждениях выше надо просто заменить R на S и r на s).

Теперь докажем, что

$$\pi_{B_1,\ldots,B_u}(Q) \supset \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S). \tag{63}$$

По условию, объединение в правой части не должно содержать дубликатов (в предположении, что $\pi_{B_1,\ldots,B_u}(R)$ и $\pi_{B_1,\ldots,B_u}(S)$ не содержат дубликатов). Такое объединение без дубликатов может быть вычислено следующим образом:

$$\tilde{S} = \pi_{S.*}\left(R \underset{R.B_1=S.B_1\wedge\ldots\wedge R.B_u=S.B_u}{\bowtie} S\right); \tag{64}$$

$$\overline{S} = \pi_{S.*}\left(\tilde{S} \underset{\tilde{S}.A\neq S.A}{\bowtie} S\right); \tag{65}$$

$$\pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S) = \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(\overline{S}). \tag{66}$$

Поэтому условие (63) эквивалентно условию

$$\pi_{B_1,\ldots,B_u}(Q) \supset \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(\overline{S}). \tag{67}$$

Покажем, что последнее истинно. Пусть $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(\overline{S})$. Тогда $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R)$, либо $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(\overline{S})$. Предположим сначала, что $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R)$. Это означает, что существует $r \in R$ такой, что

$$r.B_1 = b_1,\ldots,r.B_u = b_u. \tag{68}$$

Положим

$$r.A = a. \tag{69}$$

По определению колоночного индекса (свойства (1) и (3)) отсюда следует что $(a,b_1) \in I_{R.B_1}$. В силу свойств доменно-интервальной фрагментации существует $i \in \mathbb{N}$ такое, что $(a,b_1) \in I^i_{R.B_1}$. Тогда с учетом (54) и (56) получаем

$$p = (a) \in P_R. \tag{70}$$

Положим

$$r' = \&_R(p.A). \tag{71}$$

По построению

$$r'.A = a. \tag{72}$$

Принимая во внимание, что A — суррогатный ключ, из (72) и (69) следует

$$r = r'. \tag{73}$$

Из (58), (70), (71), (73) следует, что $r \in Q$. Вместе с (68) это дает $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(Q)$, то есть (63) имеет место для рассмотренного случая.

Предположим теперь, что

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(\overline{S}). \tag{74}$$

Это означает, что существует $\overline{s} \in \overline{S}$ такой, что

$$\overline{s}.B_1 = b_1,\ldots,\overline{s}.B_u = b_u. \tag{75}$$

Положим

$$\overline{s}.A = a. \tag{76}$$

Из (65) следует, что $\overline{s} \in S$. По определению колоночного индекса из (75) следует что

$$\begin{gathered}(a,b_1) \in I_{S.B_1};\\ \ldots\ldots\\ (a,b_u) \in I_{S.B_u}.\end{gathered} \tag{77}$$

В силу свойств доменно-интервальной фрагментации существует i такой, что $(a,b_1)\in I^i_{S.B}$. По определению транзитивной относительно $I_{S.B_1}$ фрагментации отсюда, с учетом (77), имеем

$$(a,b_2)\in I^i_{S.B_2};$$
$$\ldots\ldots\ldots$$
$$(a,b_u)\in I^i_{S.B_u}.$$

Покажем, что

$$a\notin \tilde{P}^i \tag{78}$$

Предположим противное, то есть $a\in\tilde{P}^i$. Тогда существует $(a',b'_1,\ldots,b'_u)\in R$ такой, что $(a',a)\in\tilde{P}^i$. По определению колоночного индекса $(a',b'_1)\in I_{R.B_1}$. В силу свойств доменно-интервальной фрагментации $(a',b'_1)\in I^i_{R.B_1}$. По определению транзитивной относительно $I_{R.B_1}$ фрагментации отсюда следует

$$(a',b'_2)\in I^i_{R.B_2};$$
$$\ldots\ldots\ldots$$
$$(a',b'_u)\in I^i_{R.B_u}.$$

Но тогда с учетом (53) и (52) получаем

$$b'_1=b_1;$$
$$\ldots$$
$$b'_u=b_u.$$

Отсюда в силу (64) следует $(a,b_1,\ldots,b_u)\in\tilde{S}$. Принимая во внимание (65), получаем $(a,b_1,\ldots,b_u)\notin\overline{S}$. Так как мы предположили, что $\pi_{B_1,\ldots,B_u}(S)$ не содержит дубликатов, отсюда следует $(b_1,\ldots,b_u)\notin\pi_{B_1,\ldots,B_u}(\overline{S})$. Получили противоречие с (74). Таким образом, (78) истинно. Тогда из (55) следует, что $(a)\in P^i_S$. С учетом (57) получаем $(a)\in P_S$. Теперь из (58), (76), (75) и (65) следует, что $(a,b_1,\ldots,b_u)\in Q$, то есть $(b_1,\ldots,b_u)\in\pi_{B_1,\ldots,B_u}(Q)$. *Теорема доказана.*

2. Декомпозиция реляционных операций с использованием фрагментированных колоночных хеш-индексов

В этом разделе вводится понятие колоночного хеш-индекса. Колоночный хеш-индекс позволяет использовать один колоночный индекс для индексирования нескольких атрибутов одного отношения. Для распределенных колоночных хеш-индексов рассматривается декомпозиция операций пересечения, объединения и естественного соединения.

2.1. Колоночный хеш-индекс

Определение 1. Пусть задано отношение $R(A,B_1,\ldots,B_u,\ldots)$. Пусть задана инъективная хеш-функция $h:\mathfrak{D}_{B_1}\times\ldots\times\mathfrak{D}_{B_u}\to\mathbb{Z}_{\geq 0}$. *Колоночным хеш-индексом* $I_h(A,H)$ атрибу-

тов $B_1,\ldots,B_u$ отношения R будем называть упорядоченное отношение, удовлетворяющее тождеству:

$$I_h = \tau_H\left(\pi_{A,h(B_1,\ldots,B_u)\to H}(R)\right). \tag{79}$$

Колоночный хеш-индекс обладает следующим основным свойством:

$$\forall r', r'' \in R\left(r'.B_1 = r''.B_1 \wedge \ldots \wedge r'.B_u = r''.B_u \Leftrightarrow h\left(r'.B_1,\ldots,r'.B_u\right) = h\left(r''.B_1,\ldots,r''.B_u\right)\right). \tag{80}$$

Заметим, что обратная импликация следует из инъективности хеш-функции h. Из (80) непосредственно вытекает следующее свойство колоночного хеш-индекса:

$$\forall r', r'' \in R\left(h\left(r'.B_1,\ldots,r'.B_u\right) \neq h\left(r''.B_1,\ldots,r''.B_u\right) \Leftrightarrow r'.B_1 \neq r''.B_1 \vee \ldots \vee r'.B_u \neq r''.B_u\right).$$

Фрагментация колоночного хеш-индекса осуществляется на основе доменно-интервального принципа с помощью функции фрагментации $\varphi_{I_h} : I_h \to \{0,\ldots,k-1\}$, определенной следующим образом:

$$\forall x \in I_h\left(\varphi_{I_h}(x) = \varphi_{\mathbb{Z}_{\geq 0}}(x.H)\right), \tag{81}$$

где $\varphi_{\mathbb{Z}_{\geq 0}} : \mathbb{Z}_{\geq 0} \to \{0,\ldots,k-1\}$ — доменная функция фрагментации для домена $\mathfrak{D}_H = \mathbb{Z}_{\geq 0}$.

2.2. Декомпозиция операции пересечения

В данном разделе рассматривается декомпозиция операции пересечения вида $\pi_{B_1,\ldots,B_u}(R) \cap \pi_{B_1,\ldots,B_u}(S)$ с использование распределенных колоночных хеш-индексов. При этом предполагается, что $\pi_{B_1,\ldots,B_u}(R)$ и $\pi_{B_1,\ldots,B_u}(S)$ не содержат дубликатов.

Пусть заданы два отношения $R\left(A,B_1,\ldots,B_u\right)$ и $S\left(A,B_1,\ldots,B_u\right)$, имеющие одинаковый набор атрибутов. Пусть имеются два колоночных хеш-индекса $I_{R,h}$ и $I_{S,h}$ для атрибутов $B_1,\ldots,B_u$ отношений R и S, построенные с помощью одной и той же инъективной хеш-функции $h : \mathfrak{D}_{B_1} \times \ldots \times \mathfrak{D}_{B_u} \to \mathbb{Z}_{\geq 0}$. Пусть для этих индексов задана доменно-интервальная фрагментация степени k:

$$I_{R,h} = \bigcup_{i=0}^{k-1} I_{R,h}^i ; \tag{82}$$

$$I_{S,h} = \bigcup_{i=0}^{k-1} I_{S,h}^i . \tag{83}$$

Положим

$$P^i = \pi_{I_{R,h}^i.A\to A_R,\, I_{S,h}^i.A\to A_S}\left(I_{R,h}^i \underset{(I_{R,h}^i.H = I_{S,h}^i.H)}{\bowtie} I_{S,h}^i\right) \tag{84}$$

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

$$P = \bigcup_{i=0}^{k-1} P^i . \tag{85}$$

Положим

$$Q = \left\{\&_R\left(p.A_R\right) \mid p \in P\right\}. \tag{86}$$

Теорема 5. $\pi_{B_1,\ldots,B_u}(Q) = \pi_{B_1,\ldots,B_u}(R) \cap \pi_{B_1,\ldots,B_u}(S)$.

Доказательство. Сначала докажем, что

$$\pi_{B_1,\ldots,B_u}(Q) \subset \pi_{B_1,\ldots,B_u}(R) \cap \pi_{B_1,\ldots,B_u}(S). \tag{87}$$

Пусть

$$(a,b_1,\ldots,b_u) \in Q. \tag{88}$$

Из (86) следует, что

$$(a,b_1,\ldots,b_u)=r\in R \tag{89}$$

и

$$a=r.A\in\pi_{A_R}(P). \tag{90}$$

Отсюда следует, что существует $p\in P$ такой, что $p.A_R=a\wedge p.A_S=a'$. С учетом (85) получаем, что существует i, для которого $p\in P^i$. С учетом (84) отсюда получаем, что при некотором $\chi\in\mathbb{Z}_{\geq 0}$:

$$(a,\chi)\in I^i_{R,h}\subset I_{R,h};$$
$$(a',\chi)\in I^i_{S,h}\subset I_{S,h}.$$

Поскольку хеш-функция h является инъективной, то для нее существует обратная функция. Положим $(b'_1,\ldots,b'_u)=h^{-1}(\chi)$. Тогда по определению колоночного хеш-индекса имеем

$$(a,b'_1,\ldots,b'_u)\in R;$$
$$(a',b'_1,\ldots,b'_u)\in S.$$

Поскольку A является суррогатным ключом в R, с учетом (89) получаем

$$(a,b_1,\ldots,b_u)\in R;$$
$$(a',b_1,\ldots,b_u)\in S.$$

Следовательно $(b_1,\ldots,b_u)\in\pi_{B_1,\ldots,B_u}(R)\cap\pi_{B_1,\ldots,B_u}(S)$, и (87) имеет место.

Теперь докажем, что

$$\pi_{B_1,\ldots,B_u}(R)\cap\pi_{B_1,\ldots,B_u}(S)\subset\pi_{B_1,\ldots,B_u}(Q). \tag{91}$$

Пусть

$$(a,b_1,\ldots,b_u)=r\in R \tag{92}$$

и

$$(a',b_1,\ldots,b_u)=s\in S. \tag{93}$$

Положим $\chi=h(b_1,\ldots,b_u)$. Тогда по определению колоночного хеш-индекса

$$(a,\chi)\in I_{R,h};$$
$$(a',\chi)\in I_{S,h}.$$

В силу свойств доменно-интервальной фрагментации существует i такой, что

$$(a,\chi)\in I^i_{R,h};$$
$$(a',\chi)\in I^i_{S,h}.$$

На основе (84) и (85) отсюда получаем $(a,a')\in P^i\subset P$. Поскольку A — суррогатный ключ в R, с учетом (86) имеем $(a,b_1,\ldots,b_u)\in Q$ и следовательно $(b_1,\ldots,b_u)\in\pi_{B_1,\ldots,B_u}(Q)$. Таким образом (91) имеет место. *Теорема доказана.*

2.3. Декомпозиция операции объединения

Пусть заданы два отношения $R(A,B_1,\ldots,B_u)$ и $S(A,B_1,\ldots,B_u)$. Выполним декомпозицию операции объединения вида $\pi_{B_1,\ldots,B_u}(R)\cup\pi_{B_1,\ldots,B_u}(S)$. Пусть имеются два колоночных хеш-индекса $I_{R,h}$ и $I_{S,h}$ для атрибутов $B_1,\ldots,B_u$ отношений R и S, построенные с помощью одной и той же инъективной хеш-функции $h:\mathfrak{D}_{B_1}\times\ldots\times\mathfrak{D}_{B_u}\to\mathbb{Z}_{\geq 0}$. Пусть для этих индексов задана доменно-интервальная фрагментация степени k:

$$I_{R,h} = \bigcup_{i=0}^{k-1} I_{R,h}^i ; \tag{94}$$

$$I_{S,h} = \bigcup_{i=0}^{k-1} I_{S,h}^i . \tag{95}$$

Положим

$$P_i = \pi_A \left(I_{S,h}^i \right) \setminus \pi_{I_{S,h}^i . A} \left(I_{R,h}^i \underset{(I_{R,h}^i . H = I_{S,h}^i . H)}{\bowtie} I_{S,h}^i \right) \tag{96}$$

для всех $i = 0,\ldots,k-1$.

Определим

$$P = \bigcup_{i=0}^{k-1} P_i \ . \tag{97}$$

Положим

$$Q = \left\{ \&_S \left(p.A \right) \mid p \in P \right\} . \tag{98}$$

Теорема 6. $\pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(Q) = \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S)$.

Доказательство. Сначала докажем, что

$$\pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(Q) \subset \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S) . \tag{99}$$

Пусть

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(Q) . \tag{100}$$

Тогда как минимум одно из следующих условий истинно:

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) ; \tag{101}$$

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(Q) . \tag{102}$$

Если имеет место (101), то очевидно $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S)$ и (99) имеет место.

Пусть истинно условие (102). Тогда из (98) следует, что существует $p \in P$ такой, что

$$\&_S \left(p.A \right) = s \in S$$

и

$$s = (a, b_1,\ldots,b_u) ,$$

где $a = p.A$. Отсюда получаем $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(S)$, то есть $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S)$ и (99) также имеет место.

Теперь докажем, что

$$\pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(Q) \supset \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S) . \tag{103}$$

Пусть

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(S) . \tag{104}$$

Тогда как минимум одно из следующих условий истинно:

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) ; \tag{105}$$

$$(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(S) . \tag{106}$$

Если имеет место (105), то очевидно $(b_1,\ldots,b_u) \in \pi_{B_1,\ldots,B_u}(R) \cup \pi_{B_1,\ldots,B_u}(Q)$ и (103) имеет место.

Пусть истинно условие (106) и

$$(b_1,\ldots,b_u) \notin \pi_{B_1,\ldots,B_u}(R) . \tag{107}$$

Тогда существует

$$s = (a, b_1, \ldots, b_u) \in S\,. \tag{108}$$

Положим

$$\chi = h(b_1, \ldots, b_u)\,. \tag{109}$$

Тогда по определению колоночного хеш-индекса

$$(a, \chi) \in I_{S,h}\,.$$

В силу свойств доменно-интервальной фрагментации существует $i \in \{0, \ldots, k-1\}$ такой, что

$$(a, \chi) \in I^i_{S,h}\,.$$

Покажем, что

$$(a) \notin \pi_{I^i_{S,h}.A}\left(I^i_{R,h} \underset{(I^i_{R,h}.H = I^i_{S,h}.H)}{\bowtie} I^i_{S,h} \right) \tag{110}$$

Предположим противное, то есть

$$(a) \in \pi_{I^i_{S,h}.A}\left(I^i_{R,h} \underset{(I^i_{R,h}.H = I^i_{S,h}.H)}{\bowtie} I^i_{S,h} \right) \tag{111}$$

Тогда существует $(a') \in \pi_{I^i_{R,h}.A}\left(I^i_{R,h}\right)$ такой, что $(a', \chi) \in I^i_{R,h}$. С учетом (109) в силу инъективности хеш-функции h получаем $(a', b_1, \ldots, b_u) \in R$, то есть $(b_1, \ldots, b_u) \in \pi_{B_1, \ldots, B_u}(R)$. Получили противоречие с (107). Значит (110) имеет место. Из (110) и (96) следует, что $(a) \in P_i$, и из (97) следует, что $(a) \in P$. Тогда из (98) следует, что существует $s' = (a, b'_1, \ldots, b'_u) \in Q \subseteq S$. Так как A — суррогатный ключ, сопоставляя это с (108), получаем $(a, b_1, \ldots, b_u) = (a, b'_1, \ldots, b'_u) \in Q$, откуда $(b_1, \ldots, b_u) \in \pi_{B_1, \ldots, B_u}(Q)$. Таким образом (103) имеет место. *Теорема доказана.*

2.4. Декомпозиция операции естественного соединения

Пусть заданы два отношения

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

и

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

Пусть имеются два колоночных хеш-индекса $I_{R,h}$ и $I_{S,h}$ для атрибутов $B_1, \ldots, B_u$ отношений R и S, построенные с помощью одной и той же инъективной хеш-функции $h : \mathfrak{D}_{B_1} \times \ldots \times \mathfrak{D}_{B_u} \to \mathbb{Z}_{\geq 0}$. Пусть для этих индексов задана доменно-интервальная фрагментация степени k:

$$I_{R,h} = \bigcup_{i=0}^{k-1} I^i_{R,h}\,; \tag{112}$$

$$I_{S,h} = \bigcup_{i=0}^{k-1} I^i_{S,h}\,. \tag{113}$$

Положим

$$P^i = \pi_{I^i_{R,h}.A \to A_R,\, I^i_{S,h}.A \to A_S}\left(I^i_{R,h} \underset{(I^i_{R,h}.H = I^i_{S,h}.H)}{\bowtie} I^i_{S,h} \right) \tag{114}$$

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

$$P = \bigcup_{i=0}^{k-1} P^i\,. \tag{115}$$

Построим отношение $Q(B_1, \ldots, B_u, C_1, \ldots, C_v, D_1, \ldots, D_w)$ следующим образом:

$$Q = \{(\&_R(p.A_R).B_1,\ldots,\&_R(p.A_R).B_u,\ \&_B(p.A_B).C_1,\ldots,\&_B(p.A_B).C_v,\ \&_S(p.A_S).D_1,\ldots,\&_S(p.A_S).D_w) \mid p \in P\}. \quad (116)$$

Теорема 7. $Q = \pi_{*\setminus A}(R) \bowtie \pi_{*\setminus A}(S)$.

Доказательство. Сначала докажем, что

$$Q \subset \pi_{*\setminus A}(R) \bowtie \pi_{*\setminus A}(S). \quad (117)$$

Пусть

$$(b_1,\ldots,b_u,c_1,\ldots,c_v,d_1,\ldots,d_w) \in Q. \quad (118)$$

Из (116) следует, что существуют кортежи r и s такие, что

$$(a,b_1,\ldots,b_u,c_1,\ldots,c_v) = r \in R, \quad (119)$$

$$(a',b_1',\ldots,b_u',d_1,\ldots,d_w) = s \in S \quad (120)$$

и

$$(r.A,s.A) = p \in P. \quad (121)$$

С учетом (115) получаем, что существует i, для которого $p \in P^i$. С учетом (114) отсюда получаем, что при некотором $\chi \in \mathbb{Z}_{\geq 0}$:

$$(a,\chi) \in I^i_{R,h} \subset I_{R,h};$$
$$(a',\chi) \in I^i_{S,h} \subset I_{S,h}.$$

Поскольку хеш-функция h является инъективной, то для нее существует обратная функция. Положим $(b_1'',\ldots,b_u'') = h^{-1}(\chi)$. Тогда по определению колоночного хеш-индекса имеем

$$(a,b_1'',\ldots,b_u'') \in \pi_{A,B_1,\ldots,B_u}(R); \quad (122)$$

$$(a',b_1'',\ldots,b_u'') \in \pi_{A,B_1,\ldots,B_u}(S). \quad (123)$$

Поскольку A является суррогатным ключом в R и S, из (119), (120), (122) и (123) следует $b_1 = b_1'' = b_1',\ldots,b_u = b_u'' = b_u'$, то есть

$$(a,b_1,\ldots,b_u,c_1,\ldots,c_v) \in R,$$
$$(a',b_1,\ldots,b_u,d_1,\ldots,d_w) \in S.$$

Следовательно $(b_1,\ldots,b_u,c_1,\ldots,c_v,d_1,\ldots,d_w) \in \pi_{*\setminus A}(R) \bowtie \pi_{*\setminus A}(S)$, и (117) имеет место.

Теперь докажем, что

$$Q \supset \pi_{*\setminus A}(R) \bowtie \pi_{*\setminus A}(S). \quad (124)$$

Пусть

$$(a,b_1,\ldots,b_u,c_1,\ldots,c_v) \in R \quad (125)$$

и

$$(a',b_1,\ldots,b_u,d_1,\ldots,d_w) \in S. \quad (126)$$

Положим $\chi = h(b_1,\ldots,b_u)$. Тогда по определению колоночного хеш-индекса

$$(a,\chi) \in I_{R,h};$$
$$(a',\chi) \in I_{S,h}.$$

В силу свойств доменно-интервальной фрагментации существует $i \in \{0,\ldots,k-1\}$ такой, что

$$(a,\chi) \in I^i_{R,h};$$
$$(a',\chi) \in I^i_{S,h}.$$

На основе (114) и (115) отсюда получаем $(a,a') \in P^i \subset P$. Поскольку A — суррогатный ключ в R и S, с учетом (116), (125) и (126) имеем $(b_1,\ldots,b_u,c_1,\ldots,c_v,d_1,\ldots,d_w) \in Q$. Таким образом (124) имеет место. *Теорема доказана.*

Заключение

В статье были рассмотрены вопросы декомпозиции операций проекции, выбора, удаления дубликатов и объединения с использованием фрагментированных колоночных индексов. Предложен новый вид колоночных индексов, названных колоночными хеш-индексами. Колоночный хеш-индекс способен индексировать сразу несколько атрибутов отношения. Для распределенных колоночных хеш-индексов рассмотрена декомпозиция операций пересечения, объединения и естественного соединения. Предложенные модели и методы являются теоретической основой для разработки колоночного сопроцессора, позволяющего во взаимодействии с реляционной СУБД эффективно выполнять запросы класса OLAP.

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

Литература

1. Чернышев, Г.А. Организация физического уровня колоночных СУБД / Г.А Чернышев // Труды СПИИРАН. — 2013. — № 7. Вып. 30. — С. 204–222.
2. Abadi, D.J. The Design and Implementation of Modern Column-Oriented Database Systems / D.J. Abadi, P.A. Boncz, S. Harizopoulos, S. Idreos, S. Madden // Foundations and Trends in Databases. — 2013. — Vol. 5, No. 3. — P. 197–280. DOI: 10.1561/1900000024.
3. Abadi, D.J. Column-Stores vs. Row-Stores: How Different Are They Really? / D.J. Abadi, S.R. Madden, N. Hachem // Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 9–12, 2008, Vancouver, BC, Canada. — ACM, 2008. — P. 967–980. DOI: 10.1145/1376616.1376712.
4. Иванова, Е.В. Декомпозиция операций пересечения и соединения на основе доменно-интервальной фрагментации колоночных индексов / Е.В. Иванова, Л.Б. Соколинский // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. — 2015. — Т. 4, № 1. — С. 44–56. DOI: 10.14529/cmse150104.
5. Ivanova, E. Decomposition of Natural Join Based on Domain-Interval Fragmented Column Indices / E. Ivanova, L. Sokolinsky // Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO, May 25–29, 2015, Opatija, Croatia. — IEEE, 2015. — P. 223–226. DOI: 10.1109/mipro.2015.7160266.
6. Иванова, Е.В. Использование сопроцессоров Intel Xeon Phi для выполнения естественного соединения над сжатыми данными / Е.В. Иванова, Л.Б. Соколинский // Суперкомпьютерные дни в России: Труды международной конференции (28–29 сентября 2015 г., Москва). — М.: Изд-во МГУ, 2015. — С. 190–198.

7. Иванова, Е.В. Использование распределенных колоночных индексов для выполнения запросов к сверхбольшим базам данных / Е.В. Иванова, Л.Б. Соколинский // Параллельные вычислительные технологии (ПАВТ'2014). Труды международной научной конференции. — Челябинск: Издательский центр ЮУрГУ, 2014. — С. 270–275.
8. Иванова, Е.В. Декомпозиция операции группировки на базе распределенных колоночных индексов / Е.В. Иванова, Л.Б. Соколинский // Наука ЮУрГУ. — Челябинск: Издательский центр ЮУрГУ, 2015. — С. 15–23.
9. Иванова, Е.В. Исследование эффективности использования фрагментированных колоночных индексов при выполнении операции естественного соединения с использованием многоядерных ускорителей / Е.В. Иванова // Параллельные вычислительные технологии (ПаВТ'2015): труды международной научной конференции (30 марта — 3 апреля 2015 г., Екатеринбург). — Челябинск: Издательский центр ЮУрГУ, 2015. — С. 393–398.
10. Иванова, Е.В. Использование распределенных колоночных хеш-индексов для обработки запросов к сверхбольшим базам данных / Е.В. Иванова // Научный сервис в сети Интернет: многообразие суперкомпьютерных миров: Труды Международной суперкомпьютерной конференции (22–27 сентября 2014 г., Новороссийск). — М.: Изд-во МГУ, 2014. — С. 102–104.

Иванова Елена Владимировна, программист отдела поддержки и обучения пользователей Лаборатории суперкомпьютерного моделирования, Южно-Уральский государственный университет (Челябинск, Российская Федерация), Elena.Ivanova@susu.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 5 сентября 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 4, pp. 80–100

DOI: 10.14529/cmse150405

PARALLEL DECOMPOSITION OF RELATIONAL OPERATIONS BASED ON FRAGMENTED COLUMN INDICES

E.V. Ivanova, South Ural State University (Chelyabinsk, Russian Federation) Elena.Ivanova@susu.ru,

L.B. Sokolinsky, South Ural State University (Chelyabinsk, Russian Federation) Leonid.Sokolinsky@susu.ru

This paper is a continuation and development of the previous our work, where the decompositions of intersection and join operations for columnar indices on the basis of domain-interval fragmentation were proposed. This decomposition allows to organize the parallel execution of relational operations on the distributed columnar indices without massive data exchange between the processor nodes. In this paper the decompositions of the projection, selection, duplicate elimination and union operations are considered. Also, a new kind of columnar indices called columnar hash-indices is introduced. The columnar hash-index can index multiple attributes of a relation. For distributed columnar hash-indices, the decompositions of intersection, union, and natural join operations are considered.

Keywords: distributed column indices, domain-interval fragmentation, column hash indices, decomposition of relational operations.

References

1. Chernyshev G.A Organizaciya fizicheskogo urovnya kolonochnyh SUBD [Physical Layer Organization of Columnar DBMS] // Trudy SPIIRAN [Proceedings of the SPIIRAS]. 2013. Vol. 7. No. 30. P. 204–222.
2. Abadi D.J., Boncz P.A., Harizopoulos S., Idreos S., Madden S. The Design and Implementation of Modern Column-Oriented Database Systems // Foundations and Trends in Databases. 2013. Vol. 5, No. 3. P. 197–280. DOI: 10.1561/1900000024.
3. Abadi D.J., Madden S.R, Hachem N. Column-Stores vs. Row-Stores: How Different Are They Really? // Proceedings of the 2008 ACM SIGMOD international conference on Management of data, June 9–12, 2008, Vancouver, BC, Canada. ACM, 2008. P. 967–980. DOI: 10.1145/1376616.1376712.
4. Ivanova E.V., Sokolinsky L.B. Dekompoziciya operacij peresecheniya i soedineniya na osnove domenno-interval'noj fragmentacii kolonochnyh indeksov [Decomposition of Intersection and Join Operations Based on Domain-Interval Fragmented Column Indices] // Vestnik Yuzhno-Ural'skogo gosudarstvennogo universiteta. Seriya «Vychislitel'naya matematika i informatika» [Bulletin of South Ural State University. Series: Computational Mathematics and Software Engineering]. 2015. Vol. 4, No. 1. P. 44–56. DOI: 10.14529/cmse150104.
5. Ivanova E., Sokolinsky L. Decomposition of Natural Join Based on Domain-Interval Fragmented Column Indices // Proceedings of the 38th International Convention on In-

formation and Communication Technology, Electronics and Microelectronics, MIPRO, May 25-29, 2015, Opatija, Croatia. IEEE, 2015. P. 223–226. DOI: 10.1109/mipro.2015.7160266.

6. Ivanova E.V., Sokolinsky L.B. Ispol'zovanie soprocessorov Intel Xeon Phi dlya vypolneniya estestvennogo soedineniya nad szhatymi dannymi [Using Intel Xeon Phi coprocessor for execution of natural join on compressed data] // Superkomp'yuternye dni v Rossii: Trudy mezhdunarodnoj konferencii [Russian Supercomputer Days: Proceedings of the International Conference], September 28-29, 2015, Moscow, Russia. Moscow: MSU publishing center, 2015. P. 190–198.
7. Ivanova E.V., Sokolinsky L.B. Ispol'zovanie raspredelennykh kolonochnykh indeksov dlya vypolneniya zaprosov k sverkhbol'shim bazam dannykh [Using Distributed Column Indices for Query Execution for Very Large Databases] // Parallel'nye vychislitel'nye tekhnologii (PaVT'2014). Trudy mezhdunarodnoy nauchnoy konferentsii [Proceedings of the International Conference Parallel Computational Technologies (PCT'2014)]. Chelyabinsk: SUSU publishing center, 2014. P. 270–275.
8. Ivanova E.V., Sokolinsky L.B. Dekompoziciya operacii gruppirovki na baze raspredelennyh kolonochnyh indeksov [Decomposition of Grouping Operation Based on Fragmented Column Indices] // Nauka YUUrGU [Science of SUSU]. Chelyabinsk: SUSU publishing center, 2015. P. 15–23.
9. Ivanova E.V. Issledovanie ehffektivnosti ispol'zovaniya fragmentirovannyh kolonochnyh indeksov pri vypolnenii operacii estestvennogo soedineniya s ispol'zovaniem mnogoyadernyh uskoritelej [Research of efficiency fragmented columnar indices for the natural join operation using multi-core accelerators.] // Parallel'nye vychislitel'nye tekhnologii (PaVT'2015). Trudy mezhdunarodnoy nauchnoy konferentsii [Proceedings of the International Conference Parallel Computational Technologies (PCT'2015)], March 30 — April 3, 2015, Ekaterinburg, Russia. Chelyabinsk: SUSU publishing center, 2015. P. 393–398.
10. Ivanova E.V. Ispol'zovanie raspredelennyh kolonochnyh hesh-indeksov dlya obrabotki zaprosov k sverhbol'shim bazam dannyh [Using Distributed Column Hash Indices for Query Execution for Very Large Databases] // Nauchnyj servis v seti Internet: mnogoobrazie superkomp'yuternyh mirov: Trudy Mezhdunarodnoj superkomp'yuternoj konferencii [Scientific service in Internet: The variety of supercomputing worlds: Proceedings of the International Supercomputer Conference], September 22–27, 2014, Novorossiysk, Russia. Moscow: MSU publishing center, 2014. P. 102–104.

Received September 5, 2015.

УДК 519.688, 550.34.013.4 DOI: 10.14529/cmse150406

ТЕХНОЛОГИЯ СУПЕРКОМПЬЮТЕРНОГО 3D МОДЕЛИРОВАНИЯ СЕЙСМИЧЕСКИХ ВОЛНОВЫХ ПОЛЕЙ В СЛОЖНО ПОСТРОЕННЫХ СРЕДАХ[1]

Б.М. Глинский, В.Н. Мартынов, А.Ф. Сапетина

В работе рассматриваются вычислительные технологии решения задач, связанных с моделированием распространения сейсмических волн в неоднородных средах, характерных для вулканических структур, с использованием суперкомпьютерного моделирования в целях создания систем вибросейсмического мониторинга сейсмоопасных объектов. Построена физико-математическая модель магматического вулкана и программная реализация на основе известного численного метода, эффективно использующая архитектуру современного суперкомпьютера, оснащенного GPU. Созданы параллельные 2D и 3D алгоритмы и программы для моделирования распространения упругих волн в сложно построенной среде (2D модель есть сечение исходной 3D модели различными плоскостями и под разными углами) на основе явной конечно-разностной схемы на сдвинутых сетках и метода поглощающих границ CFS-PML. Исследована масштабируемость алгоритмов. Применение разработанной технологии позволяет гораздо эффективней проводить изучение структуры волнового поля, обусловленного геометрией внутренних границ, уточнение его кинематических и динамических характеристик.

Ключевые слова: мониторинг, 3D моделирование, упругие волны, разностные схемы, гибридный кластер, GPU.

Введение

Необходимость предсказания катастрофических явлений, которые могут быть вызваны готовящейся вспышкой вулканической деятельности, является актуальной задачей. Для решения этой задачи необходимо проведение комплексных и объективных исследований процессов, происходящих на поверхности и внутри вулканической структуры.

Одним из инструментов таких комплексных исследований является активный вибросейсмический мониторинг, при проведении которого, наряду с экспериментальными работами, требуется предварительное и сопутствующее численное моделирование распространения упругих волн в вулканических структурах.

Проведение натурных геофизических экспериментов позволяет получить некоторые представления о скоростных параметрах упругой среды, а также о геометрии изучаемого объекта. В процессе обработки результатов полевого эксперимента часто возникают сложные эффекты, требующие теоретических исследований для их интерпретации и объяснения, которые могут быть выполнены с применением масштабного численного моделирования. Таким образом, создание математических моделей изучаемых объектов и теоретическое моделирование сейсмических полей актуально и помогают в изучении реальных вулканических структур.

[1] Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии – 2015».

В связи с тем, что реальный объект исследования имеет сложный рельеф, как правило, не удается поставить площадную систему наблюдения для решения обратной задачи геофизики. Поэтому приходится решать набор прямых задач с целью определения параметров среды, соответствующих экспериментальным наблюдениям на поверхности изучаемых вулканов.

Программный комплекс, решающий поставленную задачу на основе выбранного математического аппарата, должен иметь возможность моделирования упругой неоднородной среды со сложной геометрией. В связи со сложностью и масштабом моделируемой области, решение задачи численного моделирования распространения упругих волн от сосредоточенного источника может требовать значительных вычислительных ресурсов. Поэтому необходима разработка суперкомпьютерных технологий для уменьшения времени расчета и возможностью моделирования «больших» 3D моделей упругих сред. Рассмотрим применение такой технологии на примере моделирования процесса вибросейсмического зондирования вулкана Эльбрус.

Статья организована следующим образом. В разделе 1 проводится построение геофизической модели стратовулкана Эльбрус. Разделы 2 и 3 посвящены постановке и методам решения задачи распространения сейсмических волн в упругой среде. В разделе 4 описан программный комплекс для проведения численного моделирования. Раздел 5 посвящен результатам численного моделирования для тестовой модели, построенной на основе приближенной геофизической модели вулкана Эльбрус из раздела 1. В заключении перечислены основные результаты работы и отмечены планы на будущее.

1. Геофизическая модель стратовулкана Эльбрус

Двуглавый вулкан Эльбрус (абсолютные отметки Западной и Восточной вершин 5642,7 м и 5620 м соответственно) находится на северном склоне Большого Кавказа и располагается на водоразделе рек Малки, Баксана и Кубани, впадающих в Каспийское и Черное моря соответственно. Из-за своего географического положения он как бы «нависает» над плотно заселенными районами Северного Кавказа, прилегающими территориями юга России и севера Грузии. Период образования вулкана 20 млн. лет, последнее извержение 50 год.

Наличие обширного снежно-ледового покрова делает вулкан Эльбрус еще более опасным, так как в случае будущих любой силы и типа извержений, к собственно вулканической опасности, обязательно добавится катастрофическая опасность от образования лахаров и крупномасштабных наводнений, которые могут предваряться валом воды до нескольких десятков метров [1].

На основании данных, приведенных в работах [2–6], а также в работах Н.И. Хитарова (1984), Ю.В. Нечаева (1999) и других авторов построим геофизическую модель стратовулкана Эльбрус со следующими характеристиками. Западная вершина (более старая): высота 5642 м, кратер 500–600 м, глубина 200–300 м; Восточная вершина: высота 5621 м, кратер 200 м, глубина 80 м; между вершинами 3000 м, высота седловины 5322 м; до высоты 4000 м, склоны пологие, но выше 50–600 м, средний наклон 30^0, диаметр у основания 15–18 км, относительная высота 3000 м; вулканическая постройка лежит на гранитном блоке +I (рис. 1); эффузивные породы, слагают вулканический конус +II; ниже нулевой отметки можно выделить 8 слоев (табл. 1). Зададим верхний магматический очаг в виде эллипсоида с горизонтальными и вертикальными осям 9 и 6 км

($\rho = 2{,}1$ г/см^3; $Vp = 2{,}2$ км/с); диаметр бывшего канала 130 м; зададим материнский магматический очаг: эллипсоид с горизонтальными и вертикальными осями 24 и 13 км ($\rho = 1{,}8$ г/см^3; $Vp = 1{,}9$ км/с), диаметр предполагаемого подпитывающего канала 250 м. Средний канал — цилиндр, диаметром 160 м. Итак, в качестве приближенной модели вулкана Эльбрус можно принять многослойную среду с включениями в виде эллипсов, цилиндров с параметрами, указанными в табл. 1

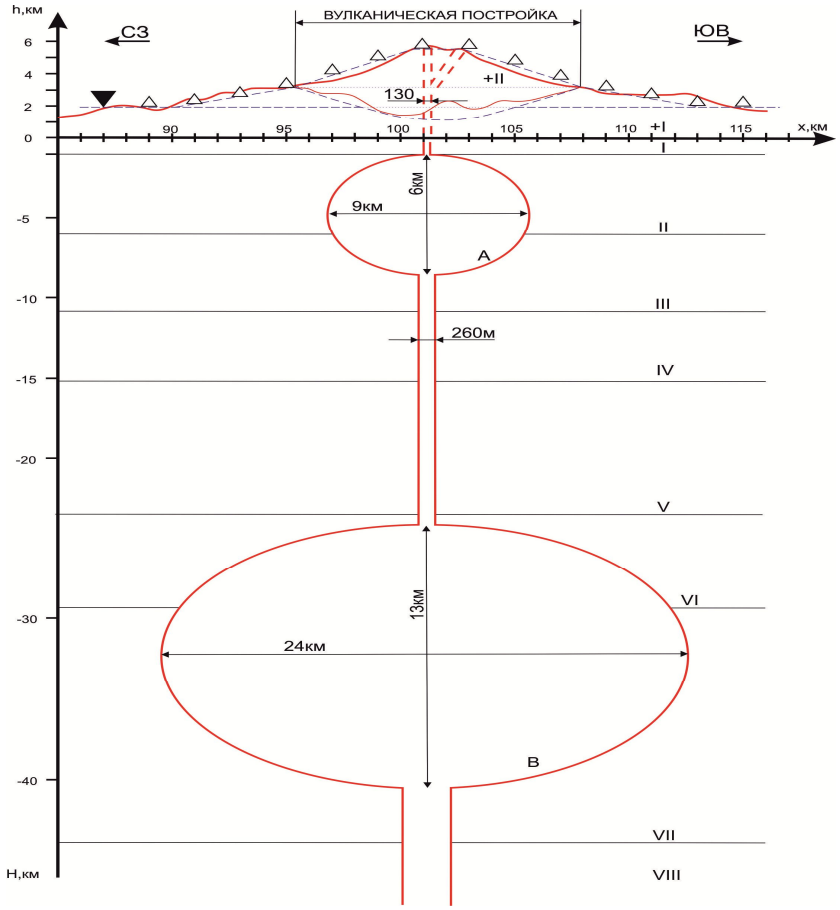


Рис. 1. Геофизическая модель вулкана Эльбрус и схема вибросейсмического мониторинга

Таблица 1

Параметры среды для геофизической модели вулкана Эльбрус

	Vp, км/с	Vs, км/с	ρ, г/см^3
Слой +II	2,85	1,65	2,4
Слой +I	3,1	1,79	2,66
Слой I	3,2	1,82	2,7
Слой II	5,9	3,42	2,85
Слой III	6,22	3,59	2,62
Слой IV	5,82	3,37	2,7
Слой V	5,97	3,45	2,75
Слой VI	6,43	3,72	2,78
Слой VII	6,95	4,03	2,81
Слой VIII	8,1	4,68	2,85

2. Постановка задачи

Численное моделирование распространения сейсмических волн в сложно построенных упругих неоднородных средах проводится на основе решения полной системы уравнений теории упругости с соответствующими начальными и граничными условиями, записанной в терминах вектора скоростей смещений $\vec{u} = (U, V, W)^T$ и тензора напряжений $\vec{\sigma} = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz})^T$.

В качестве области моделирования рассматривается изотропная 3D неоднородная сложно построенная упругая среда, представляющая собой параллелепипед, одна из граней которого является свободной поверхностью (плоскость $z=0$).

Рассмотрим прямоугольную декартову систему координат так, что ось Oz направлена вертикально вниз, а оси Ox и Oy лежат на свободной поверхности. Основные уравнения в векторной форме могут быть представлены в следующем виде:

$$\rho \frac{\partial \vec{u}}{\partial t} = [A]\vec{\sigma} + \vec{F}(t,x,y,z), \quad \frac{\partial \vec{\sigma}}{\partial t} = [B]\vec{u}, \tag{1}$$

$$A = \begin{bmatrix} \frac{\partial}{\partial x} & 0 & 0 & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} & 0 \\ 0 & \frac{\partial}{\partial y} & 0 & \frac{\partial}{\partial x} & 0 & \frac{\partial}{\partial z} \\ 0 & 0 & \frac{\partial}{\partial z} & 0 & \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix},$$

$$B = \begin{bmatrix} (\lambda+2\mu)\frac{\partial}{\partial x} & \lambda\frac{\partial}{\partial y} & \lambda\frac{\partial}{\partial z} \\ \lambda\frac{\partial}{\partial x} & (\lambda+2\mu)\frac{\partial}{\partial y} & \lambda\frac{\partial}{\partial z} \\ \lambda\frac{\partial}{\partial x} & \lambda\frac{\partial}{\partial y} & (\lambda+2\mu)\frac{\partial}{\partial z} \\ \mu\frac{\partial}{\partial y} & \mu\frac{\partial}{\partial x} & 0 \\ \mu\frac{\partial}{\partial z} & 0 & \mu\frac{\partial}{\partial x} \\ 0 & \mu\frac{\partial}{\partial z} & \mu\frac{\partial}{\partial y} \end{bmatrix}$$

где t — время,

$\rho(x,y,z)$ — плотность,

$\lambda(x,y,z)$, $\mu(x,y,z)$ — параметры Ламе.

Предполагается, что параметры упругой среды зависят от трех пространственных переменных X, Y и Z.

Начальные условия имеют вид:

$$\sigma_{xz}|_{t=0} = 0,\ \sigma_{yz}|_{t=0} = 0,\ \sigma_{xy}|_{t=0} = 0,\ \sigma_{xx}|_{t=0} = 0,\ \sigma_{yy}|_{t=0} = 0,\ \sigma_{zz}|_{t=0} = 0,$$
$$U(x,y,z)|_{t=0} = 0, V(x,y,z)|_{t=0} = 0, W(x,y,z)|_{t=0} = 0. \tag{2}$$

В качестве граничных условий на свободной поверхности выступают:

$$\sigma_{xz}|_{z=0} = 0, \sigma_{yz}|_{z=0} = 0, \sigma_{zz}|_{z=0} = 0. \tag{3}$$

В данной постановке предполагается, что правая часть (массовая сила) имеет следующий вид: $\vec{F}(t,x,y,z) = F_x\vec{i} + F_y\vec{j} + F_z\vec{k}$, где $\vec{i}$, $\vec{j}$, $\vec{k}$ — единичные направляющие векторы координатных осей.

Например, для источника типа «центр давления» получим представление: $\vec{F}(t,x,y,z) = \delta(x-x_0)\delta(y-y_0)\delta(z-z_0)f(t)$, где (x_0, y_0, z_0) координаты источника, а $\delta(x)$ — дельта функция.

3. Метод решения задачи

Метод решения поставленной задачи (1)–(3) основан на использовании конечно-разностного метода. Применяется известная конечно-разностная схема, хорошо себя зарекомендовавшая. Алгоритм построения конечно-разностной схемы наиболее подробно изложен в статьях [7–9]. Расчет сеточных коэффициентов в разностной схеме проводится на основе интегральных законов сохранения. Используемая конечно-разностная схема имеет второй порядок аппроксимации по времени и пространству [7], рассматриваются только равномерные сетки. Для примера представим несколько конечно-разностных уравнений используемой схемы:

$$\frac{\rho_{i,j,k}+\rho_{i-1,j,k}}{2}\frac{u^{n+1}_{i-\frac{1}{2},j,k}-u^{n}_{i-\frac{1}{2},j,k}}{\tau} = \frac{(\sigma^{n+\frac{1}{2}}_{xxi,j,k}-\sigma^{n+\frac{1}{2}}_{xxi-1,j,k})}{\Delta x} + \frac{(\sigma^{n+\frac{1}{2}}_{xyi-\frac{1}{2},j+\frac{1}{2},k}-\sigma^{n+\frac{1}{2}}_{xyi-\frac{1}{2},j-\frac{1}{2},k})}{\Delta y} +$$
$$\frac{(\sigma^{n+\frac{1}{2}}_{xzi-\frac{1}{2},j,k+\frac{1}{2}}-\sigma^{n+\frac{1}{2}}_{xzi-\frac{1}{2},j,k-\frac{1}{2}})}{\Delta z} + f^{n}_{xi,j,k}, \tag{4}$$

$$\frac{(\sigma^{n+\frac{1}{2}}_{xzi-\frac{1}{2},j,k-\frac{1}{2}}-\sigma^{n-\frac{1}{2}}_{xzi-\frac{1}{2},j,k-\frac{1}{2}})}{\tau} = \mu 1_{i-\frac{1}{2},j,k-\frac{1}{2}}\left(\frac{u^{n}_{i-\frac{1}{2},j,k}-u^{n}_{i-\frac{1}{2},j,k-1}}{\Delta z} + \frac{w^{n}_{i,j,k-\frac{1}{2}}-w^{n}_{i-1,j,k-\frac{1}{2}}}{\Delta x}\right) \tag{5}$$

где $\mu 1_{i-\frac{1}{2},j,k-\frac{1}{2}} = \left(\frac{1}{4}\left(\frac{1}{\mu_{i,j,k}}+\frac{1}{\mu_{i-1,j,k}}+\frac{1}{\mu_{i,j,k-1}}+\frac{1}{\mu_{i-1,j,k-1}}\right)\right)^{-1}$.

Для поглощения отражений от границ области моделирования используется вспомогательный метод CFS-PML [10–12]. Его преимущества по сравнению с классическим методом PML заключаются в том, что он дает более «качественную» картину волнового поля для данной задачи, более прост в реализации и экономичен с вычислительной точки зрения.

Поскольку область расчета представляет параллелепипед, а свободная поверхность располагается на верней его грани, то каждая из его границ, за исключением свободной поверхности, окружается поглощающим слоем. Во внутренней области волновое поле рассчитывается по первоначальным конечно-разностным уравнениям, а при попадании волны в зону поглощения происходит расчет по новым формулам с демпфирующими параметрами, описывающими подход к созданию поглощающих границ. Выбор значений демпфирующих параметров для численных расчетов в соответствующих поглощающих слоях проводится на основе результатов работы [11].

4. Программный комплекс для проведения численных расчетов

Для численного моделирования распространения сейсмических волн в моделях сред характерных для вулканических структур предполагается разработать комплекс алгоритмов и программ, позволяющий проводить конструирование модели сложной геометрии и, затем, расчеты одновременно по нескольким программам для этой модели. Он должен включать:

- программу конструирования сеточных моделей сложно построенных сред с включениями, характерными для магматических вулканов;
- программу численного моделирования распространения упругих волн в 3D неоднородных упругих средах с криволинейной свободной поверхностью;
- программу численного моделирования распространения упругих волн в 3D неоднородных упругих средах с прямолинейной свободной поверхностью;
- программу численного моделирования распространения упругих волн в 2D неоднородных упругих средах с криволинейной свободной поверхностью, для заданного сечения рассматриваемой 3D модели;
- программу численного моделирования распространения упругих волн в 2D неоднородных упругих средах с прямолинейной свободной поверхности.

Расчет может производиться одновременно для 3D модели среды и нескольких 2D сечений с целью облегчения интерпретации и получения необходимой информации об особенностях строения волнового поля для выбранной геофизической модели, системы наблюдения и положения источника. В данной реализации рассмотрен вариант алгоритмов и программ для случая прямолинейной свободной поверхности.

Разработка программ велась с учетом особенностей архитектуры гибридного кластера НКС-30Т+GPU ССКЦ ИВМиМГ СО РАН (http://www2.sscc.ru). Он состоит из 40 вычислительных узлов HP SL390s G7, каждый узел содержит два 6-ядерных CPU Xeon X5670 и три карты NVIDIA Tesla M2090 на архитектуре Fermi, у каждой 1 GPU с 512 ядрами и 6 ГБ оперативной памяти GDDR5. Суммарно НКС-30Т+GPU содержит 80 процессоров (480 ядер) CPU и 120 процессоров (61440 ядер) GPU. Пиковая производительность — 85 Тфлопс.

Программы написаны на языке программирования C++ с использованием технологий CUDA (Compute Unified Device Architecture) и MPI (Message Passing Interface).

Разработан построитель моделей среды, который позволяет конструировать на сеточном уровне сложные модели упругих сред на основе идеи Z порядка, описанной в работах [8, 9]. Предполагается, что задана крупноблочная модель среды, составленная из горизонтальных слоев, в вершинах которых задаются параметры среды (Vp, Vs и ρ). Эти параметры являются непрерывными внутри каждого блока. Разрывы проходят только по граням соседних слоев. Далее происходит интерполяция параметров среды на более «мелкую» расчетную сетку. После того как построена основная сеточная модель трехмерно-неоднородной упругой среды, возможно дальнейшее усложнение ее геометрической структуры. В построенную модель можно «вставлять» различные геометрические объекты со своими упругими параметрами среды, имеющие как численное, так и аналитическое описание.

Для расчетов разработанным программам требуется набор входных файлов с информацией о моделируемой среде и параметрами расчета в определенном формате. Входная информация содержит физические размеры модели, количество шагов дискретизации по трем пространственным направлениям и по времени, информацию о точечном источнике сейсмических волн (его несущая частота и расположение) и информацию о системе наблюдения. Результатом работы программы является набор бинарных файлов, содержащих снимки волнового поля в указанных плоскостях и теоретические сейсмограммы.

4.1. Параллельная реализация и адаптация алгоритма для решения 3D задачи

В ходе работы создана параллельная программа для численного моделирования распространения волн в трехмерных неоднородных упругих средах, реализующая указанный конечно-разностный метод и метод поглощающих слоев и рассчитанная на использование вычислительных узлов гибридного кластера. Подробное описание параллельной реализации и адаптации алгоритмов приведено в статье [13].

Для распараллеливания данной задачи используется декомпозиция области на слои вдоль направления одной из координатных осей (рис. 2). Каждый слой рассчитывается на отдельном узле, где, в свою очередь, он разбивается еще на подслои вдоль другой координатной оси, по числу графических ускорителей на узле.

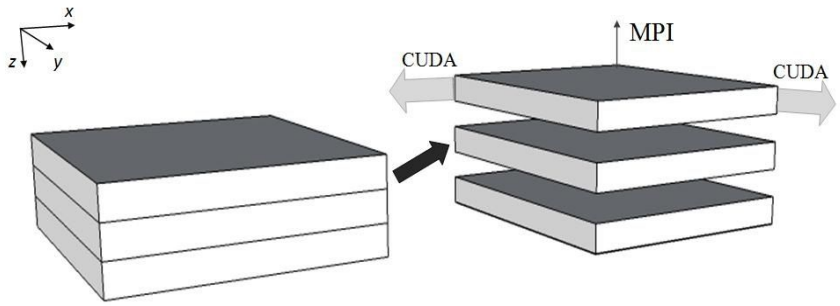


Рис. 2. Декомпозиция расчетной области для гибридного кластера

При такой реализации каждая графическая карта рассчитывает свою сеточную область внутри подслоя на каждом временном шаге независимо от других, за исключением точек, находящихся на границе между двумя соседними областями. Эти точки являются общими для каждой из областей и для продолжения счета необходимо производить обмен информацией об искомых величинах между «соседями». Обмены производятся при помощи технологии MPI. Работа с графическими ускорителями осуществляется с помощью технологии CUDA, в том числе переключение управления между ускорителями. При этом на каждой графической карте параллельная часть кода выполняется как большое количество нитей. Такой гибридный подход обеспечивает высокую степень параллелизма.

Для эффективного использования выбранной для вычислений архитектуры необходима адаптация и оптимизация используемых в ходе моделирования алгоритмов, основанная на знании архитектуры гибридного кластера и его составляющих и необходимых программных средств.

В ходе адаптации выбранных алгоритмов решена проблема выбора сетки блоков и сетки нитей в блоке для осуществления вычислений на GPU. С этим выбором естественно связаны модификации выбранных алгоритмов. Были рассмотрены два наиболее целесообразных (по степени параллелизма) варианта задания сеток и проведен анализ времени исполнения выбранных вариантов. На основе полученных выводов последующая работа велась для полностью трехмерной декомпозиции расчетной области на нити и блоки, размер которых по компоненте x, по возможности должен быть равен или хотя бы кратен длине warp'a (количество физически одновременно исполняющихся нитей на GPU).

Для реализации метода поглощающих границ для каждого поглощающего слоя добавлены свои трехмерные сетки нитей и блоков, соответствующие геометрии каждого поглощающего слоя. Это позволяет избежать увеличения времени счета, значительно сокращая количество ветвлений.

Использование явной конечно-разностной схемы на сдвинутых сетках и разбиение на слои и подслои вдоль разных координатных направлений позволяет обмениваться не всеми компонентами вектора скоростей смещений $\vec{u} = (U, V, W)^T$ и тензора напряжений $\vec{\sigma} = (\sigma_{xx}, \sigma_{yy}, \sigma_{zz}, \sigma_{xy}, \sigma_{xz}, \sigma_{yz})^T$ между каждыми слоем и подслоем, что позволяет сократить количество обменов. Также, если использовать для вычислений только карты, поддерживающие прямое копирование данных с GPU на GPU по PCI-E шине (два GPU из трех на узлах HP SL390s G7), то обмены между вычислительными узлами и графическими картами можно проводить практически параллельно, не используя во втором случае память хоста. Детальная схема обменов представлена на рис. 3.

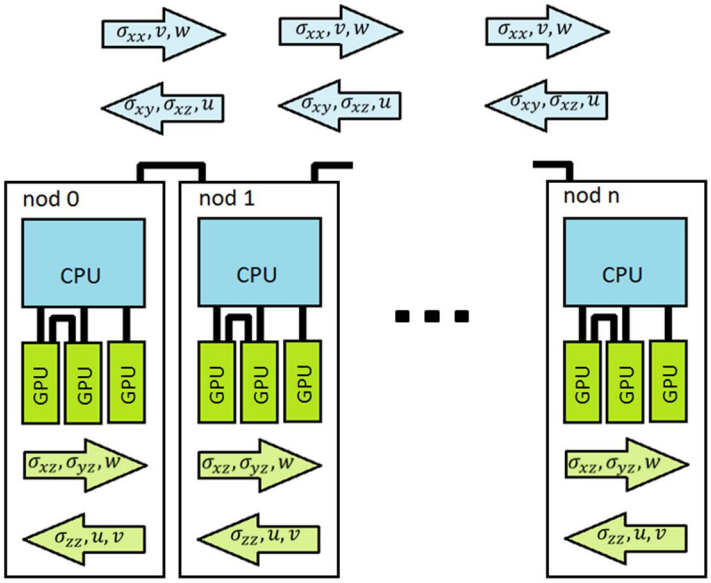


Рис. 3. Схема обменов данными между узлами и графическими ускорителями

Для уменьшения времени доступа к памяти производится оптимальное расположение используемых трехмерных массивов в глобальной памяти графической карты. Также используется константная память графической карты, в ней хранятся константы, используемые на каждом шаге по времени. Так как решаемая задача оперирует трехмерными массивами, то сложно говорить об использовании разделяемой или текстурной памяти графической карты.

4.2. Преимущества комбинирования решения 2D и 3D задач

Для проведения численного моделирования распространения сейсмических волн в двухмерной упругой среде разработано программное обеспечение на основе программы для трехмерного случая, использующее три графических ускорителя одного вычислительного узла гибридного кластера. В том числе для решения 2D задачи также используется разностная схема на сдвинутых сетках и вспомогательный метод CFS-PML в их двухмерном варианте.

Преимущество данной реализации заключается в возможности провести полномасштабный 2D расчет всего на одном вычислительном узле. Для сравнения, соответствующая 3D задача потребует для расчета решения практически все ресурсы гибридного кластера и намного больше времени для вычислений. Таким образом, предварительные расчеты для характерных 2D сечений исходной 3D модели позволяют за время расчета 3D задачи получить первоначальные результаты. Для примера приведем времена расчета тестовой 3D задачи и соответствующей ей 2D задачи в табл. 2.

Таблица 2

Время расчета тестовых 3D и 2D задач

Задача	**3D**	**2D**
Кол-во узлов расчетной сетки (X×Y×Z×T)	$500^3 \times 1000$	$500^2 \times 1000$
Время расчета	347,2 с	1,49 с

Такой подход значительно ускоряет процесс исследования особенностей волнового поля при моделировании процесса вибросейсмического зондирования вулканической постройки и позволяет провести меньшее количество расчетов для трехмерной задачи, требующих значительно большего числа ресурсов. Комбинирование расчетов 2D и 3D задач позволяет наиболее оптимально использовать архитектуру гибридного кластера.

Построение модели и реальный расчет волнового поля на узле с тремя графическими ускорителями для соответствующих сеток по времени и пространству (6000×9000 узлов по пространству и 25000 шагов по времени) занимает всего 12 минут.

4.3. Исследование времени работы параллельных программ

Для анализа производительности разработанного программного обеспечения исследована его сильная и слабая масштабируемость. Под сильной масштабируемостью будем понимать уменьшение времени счета одного шага одной и той же задачи при использовании большего числа графических ядер в рамках одной графической карты. Ее исследование позволяет понять насколько эффективно применяемые алгоритмы используют архитектуру самой графической карты. Под слабой масштабируемостью будем понимать сохранение времени счета одного шага одного и того же объема задачи при одновременном увеличении количества графических карт.

Слабая масштабируемость исследована для двух вариантов реализации разработанного программного обеспечения для решения 3D задачи, которые направлены на использование распространенных гибридных архитектур, представленных в России. В первом варианте на каждом узле используется две графических карты, поддерживающие прямое копирование данных с GPU на GPU по PCI-E шине, что позволяет, как было сказано выше, практически параллельно производить обмены между вычислитель-

ными узлами и графическими картами. Во втором случае реализации для вычислений используются три карты на каждом вычислительном узле и обмены между GPU проводятся с использованием памяти хоста.

Результаты исследований для программ 3D моделирования приведены на рис. 4, 5 в виде графиков. Из графика на рис. 4 видно, что задача хорошо ложится на архитектуру графической карты (получено ускорение около 40 раз при использовании всех ядер GPU по сравнению с одним ядром GPU).

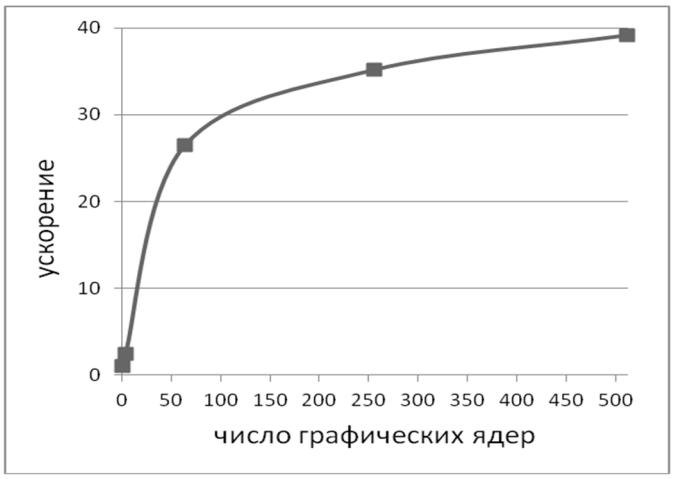
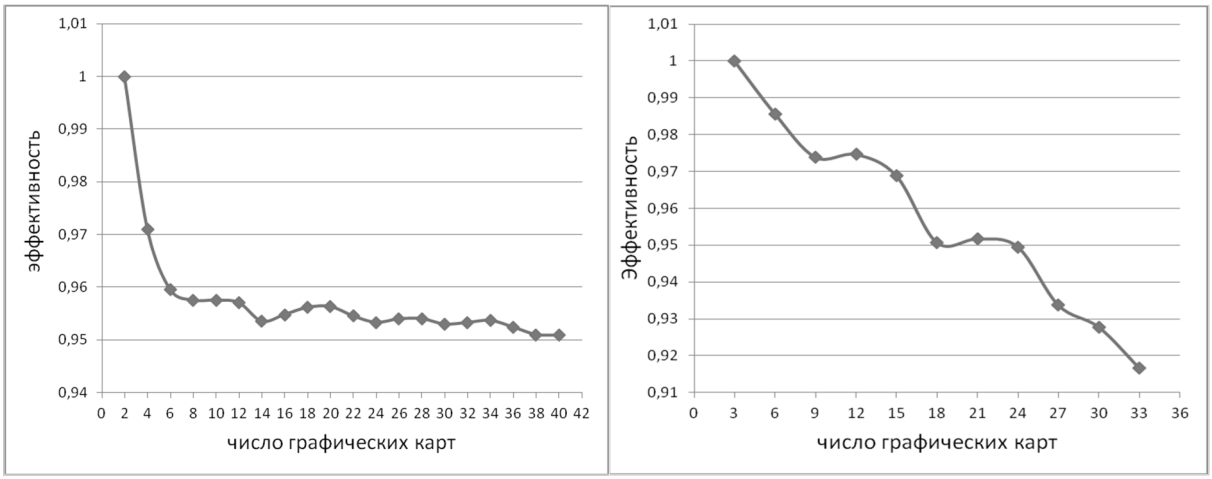


Рис. 4. График исследования сильной масштабируемости



а) использование двух GPU на узле б) использование трех GPU на узле

Рис. 5. Графики исследования слабой масштабируемости

Из графиков на рис. 5 видно, что эффективность (отношение времени расчета на n узлах в n раз большей задачи к времени расчета на 1 узле исходной задачи) для варианта реализации с тремя GPU на узле падает быстрее, так как процесс обменов не удается распараллелить, в отличии от случая с двумя GPU на узле. Но в целом достигнута эффективность около 92 % при увеличении количества графических карт до 30 штук, что говорит о действенности предложенного процесса обменов между графическими картами и узлами при расчете.

Сравним время, затраченное на численное моделирование 3D модели с использованием для вычислений узлов с GPU, с аналогичным временем для расчета на классическом кластере с CPU. Используем время полномасштабного расчета, проведенного на

вычислительных блэйд-серверах hp ProLiant BL2x220c G5, находящихся в составе НКС-30Т ССКЦ ИВМиМГ СО РАН, приведенное в работе [8]. Параметры расчета указаны в табл. 3. Соответствующее время составило 31 ч 15 мин 17 с (112517 с).

Аналогичное время расчета с теми же размерами расчетной сетки на узлах гибридного кластера с помощью разработанного программного обеспечения составило 2 ч 56 мин (10560 с); полученное ускорение составило 10,66 раза (табл. 4).

Таблица 3

Параметры расчета

Кол-во узлов расчетной сетки	по оси X	1677
	по оси Y	1059
	по оси Z	971
	по времени	10313

Таблица 4

Сравнение времени расчетов на гибридном кластере и кластере с классической MPP-архитектурой

	Классический MPP кластер	Гибридный кластер
Вид узлов кластера	HP BL2x220c G5	HP SL390s G7
Количество узлов, использованных для моделирования	20	15
Количество используемых ядер	160 CPU ядер	15360 GPU ядер
Время, затраченное на моделирование	31 ч 15 мин 17 с (112517 с)	**2 ч 56 мин** (10560 с)
Ускорение		в 10,66 раза

5. Результаты численного моделирования

Для иллюстрации работы разработанного комплекса программ численного моделирования распространения сейсмических волн в сложно построенных средах проведены тестовые расчеты, которые продемонстрировали возможности комплекса и проблемы, возникающие при визуализации и интерпретации результатов расчета.

Для расчета в качестве среды взят фрагмент исходной приближенной модели стратовулкана Эльбрус, описанной в главе 1 и включающий в себя только верхнюю магматическую камеру и прилегающие к ней каналы, расположенные в пятислойной среде. Параметры среды взяты из табл. 1 (слои +I–IV) и описания магматических включений. Система возбуждения состоит из точечного источника типа «центр давления» с частотой 8 Гц, располагающимся вблизи свободной поверхности в левой части расчетной области.

Результаты 3D численного моделирования содержат большой объем информации. В связи с этим, для представления и анализа результатов численного моделирования мы используем теоретические сейсмограммы и мгновенные снимки различных сечений 3D волнового поля плоскостями, проходящие через линии возможной системы наблюдений.

На рис. 6 представлены снимки волнового поля в плоскости, проходящей через точечный источник и ось симметрии верхнего канала.

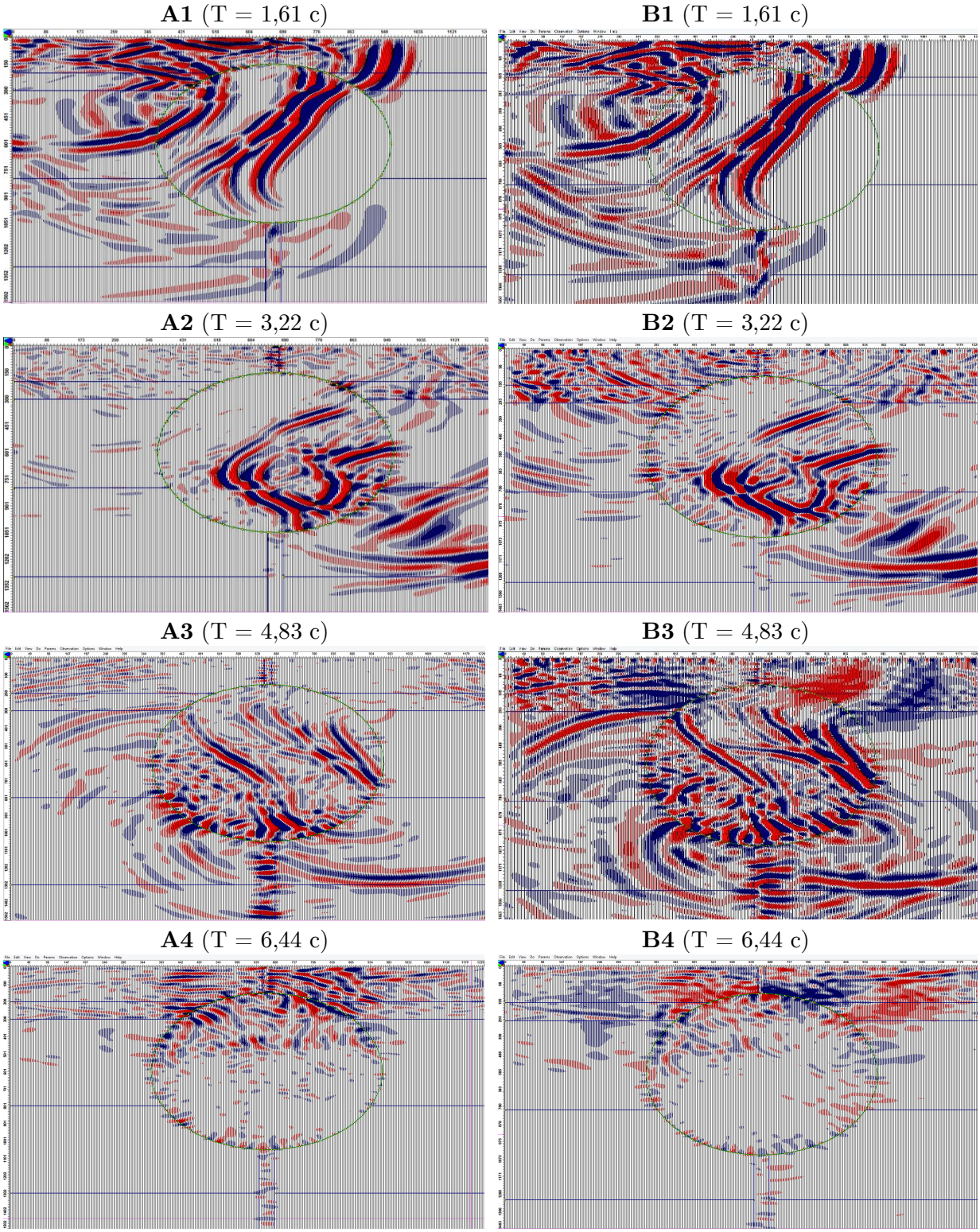


Рис. 6. Мгновенные снимки сечения 3D волнового поля в различные моменты времени, компонента U, плоскость XZ. A1–A4 снимки полученные в ходе 3D моделирования, B1–B4 снимки полученные в ходе 2D моделирования

Слева буквами A1–A4 обозначены мгновенные снимки, полученные в ходе решения 3D задачи, а справа буквами B1–B4 обозначены снимки, полученные в ходе 2D моделирования. Визуализация снимков проведена с помощью программы Aspis, разработанной

в ОАО «Сибнефтегеофизика». Фактически, данные снимки являются иллюстрацией нашего предположения о том, что 2D моделирование может быть использовано для возможного построения 3D моделей среды.

Из приведенных мгновенных снимков видно, что волновое поле имеет сложную картину и существенно зависит от геометрии, размеров и свойств эллиптических включений. Для проведения геофизической интерпретации полученных теоретических сейсмограмм, даже для этого фрагмента исходной приближенной модели вулкана Эльбрус, потребуется проведение большой серии вычислительных экспериментов. Для слоистой среды, содержащей включения, у которых может меняться ориентация и форма их границ (например, присутствие в модели пересечения нескольких объектов различной природы под различными углами), обусловленные приближением к реальной модели среды, задача значительно усложняется. Тем не менее, представленные эксперименты показывают, что численное моделирование может дать значительную информацию для организации проведения натурных экспериментов и интерпретации результатов экспериментальных наблюдений, полученных в ходе вибросейсмического мониторинга.

Заключение

В работе предложена суперкомпьютерная технология решения задачи распространения упругих волн в сложно построенных средах, учитывающая различные по размерности численные методы исследования неоднородной среды и архитектуру суперкомпьютера.

Предложен комплекс параллельных алгоритмов и программ для гибридного кластера с графическими ускорителями, реализующий распространения упругих волн в 2D и 3D моделях в сложно построенных средах. Причем, фактически созданы две программы, позволяющие проводить расчеты на гибридных кластерах как с двумя, так и стремя графическими картами на узле.

Проведенные численные эксперименты для упрощенной модели вулкана Эльбрус показывают, что волновое поле от точечного источника имеет сложную картину и существенно зависит от геометрии, размеров и свойств эллиптических включений. Особенности распространения упругих волн в средах такого типа могут быть использованы при вибросейсмическом мониторинге вулканических структур. Проведение серии расчетов позволяет выбрать модель вулканической структуры адекватную результатам натурных наблюдений по кинематическим и динамическим характеристикам.

В дальнейшем предполагаются развитие предложенной суперкомпьютерной технологии численных расчетов и последующее проведение численных экспериментов для моделей с криволинейной свободной поверхностью.

Работа поддержана проектом РФФИ №13-07-00589.

Литература

1. Лаверов, Н.П. Новейший и современный вулканизм на территории России. / Н.П. Лаверов [и др.] — М.: Наука, 2005. — 604 с.
2. Мясников, А.В. Мониторинг состояния магматических структур вулкана Эльбрус по наблюдениям литосферных деформаций баксанским лазерным интерферометром [Текст] : дис. / А.В. Мясников — Москва, 2012.

3. Гурбанов, А.Г. Активный вулкан Эльбрус и этапы его геологической истории / А.Г. Гурбанов, В.М. Газеев, О.А. Богатиков [и др.] // Современные методы геолого-геофизического мониторинга природных процессов на территории Кабардино-Балкарии. — 2005. — С. 94–119.
4. Авдулов, М.В. О геологической природе гравитационной аномалии Эльбруса / М.В. Авдулов, Н.В Короновский // Изв. АН СССР. Сер.геолог. — 1962. — № 9. — С. 67–74.
5. Авдулов, М.В. О геологической природе Эльбрусского гравитационного минимума / М.В. Авдулов // Вестник МГУ. Сер. 4. Геология. — 1993. — № 3. — С. 32–39.
6. Собисевич, А.Л. Избранные задачи математической геофизики, вулканологии и геоэкологии. / А.Л. Собисевич — М.: ИФЗ РАН, 2012. — Т. 1. — 512 с.
7. Bihn, M. Stable Discretization Scheme for the Simulation of Elastic Waves / M. Bihn, T.A. Weiland // Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics (IMACS 1997). — 1997. — Vol. 2. — P. 75–80.
8. Глинский, Б.М. Численное моделирование и экспериментальные исследования грязевого вулкана «Гора Карабетова» вибросейсмическими методами / Б.М. Глинский, Д.А. Караваев, В.В. Ковалевский, В.Н. Мартынов // Вычислительные методы и программирование. — 2010. — Т. 11. — С. 95–104.
9. Караваев, Д.А. Параллельная реализация метода численного моделирования волновых полей в трехмерных моделях неоднородных сред / Д.А. Караваев // Вестник Нижегородского университета им. Н. И. Лобачевского. — 2009. — № 6(1). — С. 203–209.
10. Drossaert, F. Complex frequency shifted convolution PML for FDTD modeling of elastic waves / F. Drossaert, A. Giannopoulos // Wave Motion. — 2007. — Vol. 44. — P. 593–604. DOI: 10.1016/j.wavemoti.2007.03.003.
11. Komatitsch, D. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation / D. Komatitsch, R. Martin // GEOPHYSICS. — 2008. — Vol. 73, No. 4. — P. T51–T61. DOI: 10.1190/1.2939484.
12. Hastings, F.D. Application of the perfectly matched layer (PML) absorbing boundary condition to elastic wave propagation / F.D. Hastings, J.B. Schneider, S.L. Broschat // J. Acoust. Soc. Am. — 1996. — Vol. 100(5). — P. 3061–3069. DOI: 10.1121/1.417118.
13. Сапетина, А.Ф. Численное моделирование распространения сейсмических волн в сложно построенных средах на гибридном кластере / А.Ф. Сапетина // Проблемы прочности и пластичности. — 2014. — Т. 76, № 4. — С. 288–296.

Глинский Борис Михайлович, д.т.н., заведующий лабораторией, Сибирский суперкомпьютерный центр Института вычислительной математики и математической геофизики СО РАН, заведующий кафедрой вычислительных систем, Новосибирский государственный университет (Новосибирск, Российская Федерация), gbm@opg.sscc.ru.

Мартынов Валерий Николаевич, с.н.с. лаборатории численного моделирования сейсмических полей, Институт вычислительной математики и математической геофизики СО РАН (Новосибирск, Российская Федерация), vnm@nmsf.sscc.ru.

Сапетина Анна Федоровна, аспирант, инженер лаборатории сибирского суперкомпьютерного центра, Институт вычислительной математики и математической геофизики СО РАН (Новосибирск, Российская Федерация), afsapetina@gmail.com.

Поступила в редакцию 15 апреля 2015 г.

Bulletin of the South Ural State University
Series "Computational Mathematics and Software Engineering"
2015, vol. 4, no. 4, pp. 101–116

DOI: 10.14529/cmse150406

TECHNOLOGY OF SUPERCOMPUTER SIMULATION OF SEISMIC WAVE FIELDS IN COMPLICATED MEDIA

B.M. Glinskiy, Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of Russian Academy of Sciences (Novosibirsk, Russian Federation) gbm@opg.sscc.ru,

V.N. Martynov, Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of Russian Academy of Sciences (Novosibirsk, Russian Federation) vnm@nmsf.sscc.ru,

A.F. Sapetina, Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of Russian Academy of Sciences (Novosibirsk, Russian Federation) afsapetina@gmail.com

The paper considered computing technology solving problems related to the modeling of seismic wave propagation in inhomogeneous media typical of volcanic structures using supercomputer simulations in order to create systems of vibroseis monitoring for quake-prone objects. The physico-mathematical model of the magmatic volcano is constructed and software implementation on the basis of the known numerical method that effectively using the architecture of modern supercomputers equipped with GPU is developed. The parallel 2D and 3D algorithms and software for simulation of elastic wave propagation in a complicated medium (2D model is separation of original 3D model using various angles and planes) on basis of the explicit finite-difference scheme for the shifted grids and CFS-PML method of absorbing boundaries is developed. Scalability of algorithms is investigated. The application of the developed technology allows for much more efficient to carry out studies of the structure of the wave field due to the geometry of the internal boundaries and refinement of its kinematic and dynamic characteristics.

Keywords: monitoring, 3D simulation, elastic waves, finite difference schemes, hybrid cluster, GPU.

References

1. Laverov N.P., et al. Noveyshiy i sovremennyy vulkanizm na territorii Rossii [Modern and Holocene volcanism in Russia]. Moscow, Science, 2005. 604 p.
2. Myasnikov A.V. Monitoring sostoyaniya magmaticheskikh struktur vulkana El'brus po nablyudeniyam litosfernykh deformatsiy baksanskim lazernym interferometrom [Monitoring the State of the Magmatic Structures of Elbrus Volcano Based on Observation of

Lithosphere Strains by observation of the Baksan Laser Interfrometer]. dissert. Moscow, 2012.

3. Gurbanov A.G., Gazeev V.M., Bogatikov O.A. and etc. Aktivnyy vulkan El'brus i etapy ego geologicheskoy istorii [Active volcano Elbrus and the stages of its geological history] // Modern methods of geological and geophysical monitoring of natural processes on the territory of Kabardino-Balkaria. 2005. P. 94–119.
4. Avdulov M.V., Koronovskiy N.V. O geologicheskoy prirode gravitatsionnoy anomalii El'brus [About the biological nature of Elbrus gravity anomaly] // The news of AS USSR. Geology Series. 1962. No. 9. P. 67–74.
5. Avdulov M.V. O geologicheskoy prirode El'brusskogo gravitatsionnogo minimuma [About the biological nature of the Elbrus gravity minimum] // Moscow University Geology Bulletin. 1993. No. 3. P. 32–39.
6. Sobisevich A.L. Izbrannye zadachi matematicheskoy geofiziki, vulkanologii i geoekologii [Selected problems of Mathematical Geophysics and Volcanology and Geoecology]. Moscow, IPE RAS, 2012. Vol. 1. 512 p.
7. Bihn M., Weiland T.A. Stable Discretization Scheme for the Simulation of Elastic Waves // Proceedings of the 15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics (IMACS 1997). 1997. Vol. 2. P. 75–80.
8. Glinskiy B.M., Karavaev D.A., Kovalevskiy V.V., Martynov V.N. Chislennoe modelirovanie i eksperimental'nye issledovaniya gryazevogo vulkana «Gora Karabetova» vibroseysmicheskimi metodami [Numerical modeling and experimental research of the «Karabetov Mountain» mud volcano by vibroseismic methods] // Numerical Methods and Programming. 2010. Vol. 11. P. 95–104.
9. Karavaev D.A. Parallel'naya realizatsiya metoda chislennogo modelirovaniya volnovykh poley v trekhmernykh modelyakh neodnorodnykh sred [Parallel implementation of wave field numerical modeling method in 3D models of inhomogeneous media] // Vestnik of Lobachevsky State University of Nizhni Novgorod. 2009. No. 6(1). P. 203–209.
10. Drossaert F., Giannopoulos A. Complex frequency shifted convolution PML for FDTD modeling of elastic waves // Wave Motion. 2007. Vol. 44. P. 593–604. DOI: 10.1016/j.wavemoti.2007.03.003.
11. Komatitsch D., Martin R. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation // GEOPHYSICS. 2008. Vol. 73, No. 4. P. T51–T61. DOI: 10.1190/1.2939484.
12. Hastings F.D., Schneider J.B., Broschat S.L. Application of the perfectly matched layer (PML) absorbing boundary condition to elastic wave propagation // J. Acoust. Soc. Am. 1996. Vol. 100(5). P. 3061–3069. DOI: 10.1121/1.417118.
13. Sapetina A.F. Chislennoe modelirovanie rasprostraneniya seysmicheskikh voln v slozhno postroennykh sredakh na gibridnom klastere [Numerical simulation of seismic wave propagation in a complicated medium on the hybrid cluster] // Problems of Strength and Plasticity. 2014. Vol. 76, No. 4. P. 288–296.

Received April 15, 2015.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан 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; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» (№421).

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.**
2. Адрес редакции научного журнала «Вестник ЮУрГУ», серия «Вычислительная математика и информатика»:

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, факультет ВМИ,
   кафедра ИТ, ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**