ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2013
Т. 2, № 1

ISSN 2305-9052

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

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

Основной целью издания является пропаганда научных исследований в следующих областях:

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

BULLETIN

OF THE SOUTH URAL STATE UNIVERSITY

2013
Vol. 2, no. 1

ISSN 2305-9052

SERIES
«COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING»

South Ural State University

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

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

Содержание

Contents

УДК 004.382.2

ИССЛЕДОВАНИЕ МАСШТАБИРУЕМОСТИ ПРОГРАММ С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТОВ АНАЛИЗА ПАРАЛЛЕЛЬНЫХ ПРИЛОЖЕНИЙ НА ПРИМЕРЕ МОДЕЛИ АТМОСФЕРЫ NH3D[1]

А.С. Антонов, А.М. Теплов

Рассмотрен подход к исследованию масштабируемости параллельных приложений с использованием средств анализа их работы. Для изучения масштабируемости в описанной методике используется наряду с профилированием приложения анализ его узких мест с помощью трассировщика. Приведен краткий обзор показателей эффективности работы параллельной программы, обзор подходов к изучению масштабируемости параллельных программ и инструментов для исследования параллельных приложений; описание методики исследования масштабируемости программы, а также детальное описание программы, которая использовалась для отработки методики. Приведены результаты исследования описанной параллельной программы с использованием профилирования работы и изучения трассировщиком. Для этих целей в качестве трассировщика был выбран Intel Trace Analyzer and Collector. В заключительной части сделаны выводы о применимости использованных инструментов анализа работы параллельных приложений для исследования масштабируемости.

Ключевые слова: масштабируемость, анализ эффективности, трассировка, профилирование, инструментарий для анализа параллельных приложений.

Введение

Параллельное программирование всегда связано с рядом сложностей, с которыми не сталкиваются программисты, создающие последовательные программы. Но только параллельные вычисления позволяют добиваться максимальной производительности, которая так необходима во многих современных научных расчетах. Современный численный эксперимент, проводимый на огромном суперкомпьютере, генерирует огромные объемы данных и может проходить до нескольких суток.

Вопрос рационального использования вычислительных систем, способных справиться с такими вычислениями, стоит очень остро. Он напрямую связан с масштабируемостью – свойством параллельных программ, характеризующим зависимость показателей эффективности работы параллельной программы от числа использованных процессоров. В данной статье описана методика исследования масштабируемости параллельной программы с использованием инструментов для анализа параллельных приложений и использовании полученных данных для построения картины масштабируемости приложения на примере атмосферной модели NH3D.

1. Масштабируемость параллельных программ

Для оценки качества параллельной программы введен ряд показателей [1], например:

Ускорение (speedup) $S = T_1/T_p$, где T_p — время исполнения распараллеленной программы на p процессорах, T1 — время исполнения исходной последовательной программы.

[1] Статья рекомендована к публикации программным комитетом Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: поиск новых решений – 2012».

Вычислительная сложность задачи W — количество основных вычислительных шагов лучшего последовательного алгоритма, необходимых для решения задачи на одном процессоре. W является некоторой функцией от размера входных данных программы. Если для простоты предположить, что каждый основной вычислительный шаг выполняется за единицу времени, то получим $W = T_1$.

Эффективность $E = S/p$ определяет среднюю долю времени выполнения параллельного алгоритма, в течение которого процессоры реально используются для решения задачи.

Стоимость (cost) вычислений $C = pT_p$.

$T_0 = pT_p - T_1$ — суммарные накладные расходы (total overhead). При увеличении p значение, как правило, возрастает.

Если T_1 фиксировано, то при увеличении числа процессоров p эффективность E, как правило, уменьшается за счет роста накладных расходов T_0. Получение большого ускорения за счет большого числа процессоров обычно приводит к снижению эффективности. А если фиксировано число процессоров p, то эффективность E зачастую можно повысить, увеличивая вычислительную сложность решаемой задачи W (а значит, и время T_1).

Отношение ускорения или эффективности к объему используемых ресурсов или размеру задачи характеризует то, насколько рационально программа способна использовать параллельную вычислительную систему. Для этого вводится понятие масштабируемости параллельной программы. Масштабируемость – свойство программы, определяющее зависимость её ускорения или эффективности, получаемых на вычислительной системе, от объема использованных для этого ресурсов и размера задачи. На основании данных о масштабируемости можно сделать выводы о необходимости оптимизации программы и причинах возникающих проблем.

Масштабируемость параллельных программ можно подразделить на несколько основных типов:

• Сильная масштабируемость (strong scaling) — зависимость производительности от количества процессоров p при фиксированной вычислительной сложности задачи ($W = const$).

• Масштабируемость вширь (wide scaling) – зависимость производительности от вычислительной сложности задачи W при фиксированном числе процессоров (p=const)

• Слабая масштабируемость (weak scaling) — зависимость производительности от количества процессоров p при фиксированной вычислительной сложности задачи в пересчете на один узел ($W/p = const$).

2. Подходы к изучению масштабируемости

Существует несколько подходов к изучению масштабируемости программ. Методы исследования масштабируемости по способу получения данных можно разделить на две группы: аналитические [2, 4] и эмпирические [3].

Эмпирический подход – это анализ работы реальной программы, основывающийся на данных, полученных при многократном выполнении программы на конкретной вычислительной системе. Этот подход используется наиболее часто.

Аналитический подход – это получение необходимых характеристик программы с помощью моделирования хода её выполнения по заданному принципу на менее мощной вычислительной системе и построения прогноза масштабируемости данной программы на целевой вычислительной системе.

Аналитический подход подразумевает один из видов моделирования:

– алгебраический анализ;

– синтаксический разбор исходного текста программы;

– абстрактная интерпретация;

– симулирование выполнения.

При комбинировании методов эмпирического и аналитического исследования получают так называемые смешанные методы исследования масштабируемости.

Из смешанных методов выделяют квазианалитический метод, основанный на моделировании объема вычислений на каждый узел и запуске на целевой системе. Такой подход требует большего внимания к моделированию коллективных операций. Это связано с тем, что операции типа точка-точка на целевой конфигурации компьютера будут выполняться столько же времени, а время выполнения коллективных обменов будет зависеть от числа реально участвующих процессов.

Построение модели программы возможно только при условии полного и детального знания алгоритма работы и реализации. Для построения качественной модели требуется использование специального инструментария и аккуратный перенос на модель всех деталей работы программы.

3. Категории инструментов для исследования эффективности и масштабируемости параллельных приложений

Существующие инструменты для исследования масштабируемости программ можно разделить на несколько групп:

• инструменты для анализа показателей метрик эффективности работающей программы (профилировщики, трассировщики, средства пакетного запуска, средства для инструментирования кода);

• инструменты для моделирования работы реального приложения (утилиты, моделирующее время выполнения на основе реальных трасс, исходного кода, ручного сбора данных, других моделей);

• инструменты для моделирования потенциальной масштабируемости разрабатываемого приложения (моделируют масштабируемость разрабатываемого приложения на предполагаемой архитектуре).

Профилирование – процесс получения данных, которые обобщают поведение программы: сколько раз была вызвана некоторая функция; время выполнения функции; дерево или граф вызовов в программе; значения аппаратных счетчиков и т.п. Таким образом, профилирование помогает определить «узкие места» и «горячие точки» в программе.

Профилирование может реализовываться через:

• семплирование – периодические прерывания ОС или прерывания по значению аппаратных счетчиков;

• инструментирование – вставка в программу дополнительного кода, измеряющего производительность.

Сбор трасс подразумевает запись потока событий: вход/выход в какой-то участок кода (функцию, цикл и др.); взаимодействие процессов (нитей) — прием/передача сообщений и т.д., запись того, в каком процессе/нити и когда случилось каждое событие.

Трасса позволяет анализировать производительность и корректность программы с точки зрения того, какие процессы происходят в каждый конкретный момент её выполнения.

На данный момент отсутствует инструмент, позволяющий анализировать масштабируемость приложения. Поэтому для исследования необходимо использовать сразу нескольких инструментов, которые бы позволили проводить анализ. В данной статье рассматривается связка профилировщика и трассировщика. Профилировщик необходим для определения участков кода с проблемами масштабируемости. Трассировщик необходим для установления причин проблем и путей возможного решения. Такая связка инструментов позволяет не только получить данные о масштабируемости как всего приложения, так и его участков, но и данные об особенностях его работы, а также рекомендации по эффективному запуску приложения.

Существует довольно большой набор инструментов, позволяющих проводить различные типы анализа работы приложения. Профилировку приложения можно проводить как вручную, измеряя время работы отдельных участков кода программы прямо в коде, так и любыми специализированными профилировщиками. Из множества профилировщиков хотелось бы выделить класс профилировщиков, способных собирать данные, не зависящие от текущей архитектуры, на которой выполняется приложение. Профилировщики такого класса позволяют по собранным данным подобрать оптимальную архитектуру для выполнения приложения. К таким инструментам можно отнести Rogue Wave Threadspotter. Существуют также и инструменты, в которых присутствует как профилировщик, так и трассировщик с визуализатором результатов анализа. К таким инструментам можно отнести TAU Toolkit. С его помощью можно получить больше информации о работе приложения. Также стоит выделить такие инструменты как Vampir, Scalasca, Periscope и Score-P, которые интегрируются как с TAU так и друг в друга и дополняя свои результаты исследования данными другого контекста. Данные инструменты позволяют при их совместном использовании провести комплексный анализ работы параллельного приложения и не только выявить места его наименьшей эффективности, но и определить наиболее вероятные причины проблем.

Для работы был выбран инструмент для трассировки и анализа приложений Intel Trace Analyzer and Collector, хотя для этих целей подходит практически любой трассировщик со сходным функционалом.

Intel Trace Analyzer and Collector – инструмент, позволяющий проводить сбор трасс с последующим анализом визуализированной трассы. Он состоит из двух основных составляющих: Trace Collector, осуществляющий сбор трасс, и Trace Analyzer, осуществляющий визуализацию и анализ. Поддерживается сбор трасс параллельных приложений на языках C/C++, Java, Fortran, использующих MPI, явные и неявные нити (OpenMP) или гибридную модель (MPI+OpenMP). Существуют версии для Linux и Microsoft Windows для архитектур IA32, Intel64.

4. Описание методики исследования масштабируемости

Исследование масштабируемости модели проходило в несколько этапов.

На первом этапе производился сбор данных о работе приложения и о том, какие данные могут быть использованы для оценки скорости работы. Для этого по выбранным критериям собирались данные о работе приложения в начальном виде. Производился запуск задачи фиксированного размера на разном количестве используемых процессоров. Это необходимо для определения начальной картины масштабируемости приложения.

Полученные данные говорят о наличии проблем масштабируемости и необходимости более глубокого анализа.

Измерялось время выполнения различных частей программы. Изначально в силу итерационной структуры работы алгоритма обработки данных был предложен метод измерения числа выполненных итераций за фиксированное время, но от него пришлось отказаться из-за того, что на этапе инициализации модели время выполнения различных процедур отличалось довольно сильно. Это влияло на число выполненных итераций, и потому более точным оказалось измерение среднего времени итерации отдельно от времени выполнения инициализации.

Исследуемая программа запускалась на разных конфигурациях вычислительной системы. Начиная с минимальной конфигурации число использованных процессоров пошагово увеличивалось.

После каждого запуска анализировались собранная статистика работы приложения. Данные фиксировались в логе работы. Из полученных данных о времени работы вычисляли ускорение работы и эффективность. На следующем этапе приложение инструментировалось для профилирования и определения ускорения различных частей кода. Проводилась еще одна серия запусков, в которой собирались данные о работе отдельных частей программы. Те участки, время которых росло с ростом числа процессоров, рассматривались более детально на предмет причин проблем масштабируемости и путей повышения эффективности.

При использовании Intel Trace Analyzer and Collector для углубленного анализа, данные собирались схожим образом. После каждого запуска, собранная трасса конвертировалась и архивировалась для дальнейшего анализа.

По времени выполнения отдельных частей программы вычислялись их характеристики, такие как ускорение и эффективность работы.

5. Описание модели NH3D

Исследование масштабируемости параллельного приложения проводилось на примере трехмерной гидротермодинамической негидростатической модели атмосферы [5], разрабатываемой в НИВЦ МГУ имени М.В. Ломоносова.

Программа написана на языке Фортран с использованием технологии параллельного программирования MPI. Общий объем кода составляет примерно 75 тысяч строк. В основе параллельной реализации модели для многопроцессорных систем с распределенной памятью лежит двумерное разбиение трехмерной расчетной области.

Модель построена на пошаговом преобразовании данных. Почти все время выполнения программного кода приходится на цикл интегрирования уравнений модели по времени main_loop в головной программной единице nh3d_main. Подпрограммы, вызываемые в этом цикле и реализующие компоненты численного алгоритма, не имеют логических ветвлений и выполняются детерминировано.

Такая структура программы позволила существенно упростить анализ производительности и профилирование, потому что на каждом шаге модели выполняются идентичные операции. Для уменьшения объемов собираемой информации число шагов модели было сведено к минимуму, который давал достаточную статистику работы.

Цель исследования заключалась в оптимизации ключевых процедур исходного кода приложения, позволяющей добиться повышения масштабируемости этих процедур на

больших конфигурациях вычислительной системы. Работа состояла из нескольких этапов классической схемы оптимизации приложения:

1. Профилирование приложения с выделением узких мест (процедур) при использовании большого количества вычислительных ядер. Профилирование проводилась на двух уровнях. На первом уровне диагностики отдельно оценивалось время выполнения приложения в целом и процедур библиотеки MPI. Минимальной конфигурацией была выбрана конфигурация с 4 процессорами. На меньших конфигурациях работали другие логические ветки кода. Для этого использовался пакет ITAC (Intel Trace Analyzer and Collector). На втором уровне производилась оценка времени выполнения отдельных процедур приложения. Для этих целей проводилось несколько типов профилирования. По результатам профилирования выбирались процедуры, время выполнения которых становилось критичным для времени выполнения одного шага модели в целом. При выборе процедур принимались во внимание особенности структуры кода, численные алгоритмы и организация MPI-обменов в данном приложении. На основе собранных данных были выявлены наиболее узкие места работы приложения и последующее более детальное инструментирование исходного кода для анализа с помощью Trace Analyzer.

2. Оптимизация выбранных процедур проводилась, исходя из полученных данных на всех этапах анализа. Так, на первом этапе анализа была выявлена проблема с выводом в файл. Модель регулярно записывала результаты в файл используя только один процесс. Время вывода было существенным даже на небольших конфигурациях вычислительной системы и не уменьшалось с ростом числа процессоров. Поэтому на больших конфигурациях время вывода в файл составляло существенную долю времени работы всей модели. Регулярность вывода не была критичной для работы алгоритма, и потому вывод в файл остался в конце всей работы, также планируется переход на параллельный вывод данных.

Второй оптимизацией, основанной на результатах профилирования, стало использование модернизированного варианта двух процедур, используемых на каждом шаге модели. Профилирование показало, что на больших конфигурациях вычислительной системы основу времени каждого шага составляло время выполнение двух процедур: MOMENT_MPI и THERMO_MPI. Процедура MOMENT_MPI решает три уравнения движения, а процедура THERMO_MPI — уравнение притока тепла и уравнения переноса излучения в атмосфере. Ускорение обеих процедур переставало расти при увеличении числа процессоров больше 25. На конфигурациях около 100 процессоров время увеличилось в 10 раз для процедуры MOMENT_MPI и в 40 раз для процедуры THERMO_MPI по сравнению со временем их выполнения на 25 процессорах. Более детальное профилирование показало, что процедуры используют адаптацию одного и того же алгоритма усреднения данных на границе области SPONGE, что и приводило к таким результатам. Отключение его использования в настройках работы модели позволило существенно уменьшить время одного шага модели и масштабируемость всего приложения в целом. После этого ускорение шага модели продолжало расти до конфигураций вычислительной системы, содержащих в полтора раза больше процессоров.

6. Исследование модели NH3D с помощью ITAC

При исследовании модели NH3D использовался пакет ITAC. На первом этапе исследования анализ трассы позволил исследовать общий характер поведения параллельной программы на вычислительной системе. В собранной трассе без пользовательского инструментирования исходного кода программы отображались два класса функций: время,

затраченное на вызовы функций MPI, и время выполнения кода приложения. Даже без дополнительного инструментирования исходного кода приложения можно определить ряд узких мест работы программы. Визуально легко определить наличие последовательных участков программы. Такие участки соответствуют выводу программы в файл, а также процессу инициализации данных приложения. Они ухудшают масштабируемость всего приложения в целом, потому что не получают никакого ускорения на любом масштабе вычислительной системы. На собранной трассе можно установить, какие функции MPI увеличивают разбалансировку работы приложения. На рис. 1 показана трасса работы приложения, где работу выполняет только один процесс. На рис. 2 показана трасса работы приложения, где работа процессов разбалансирована.

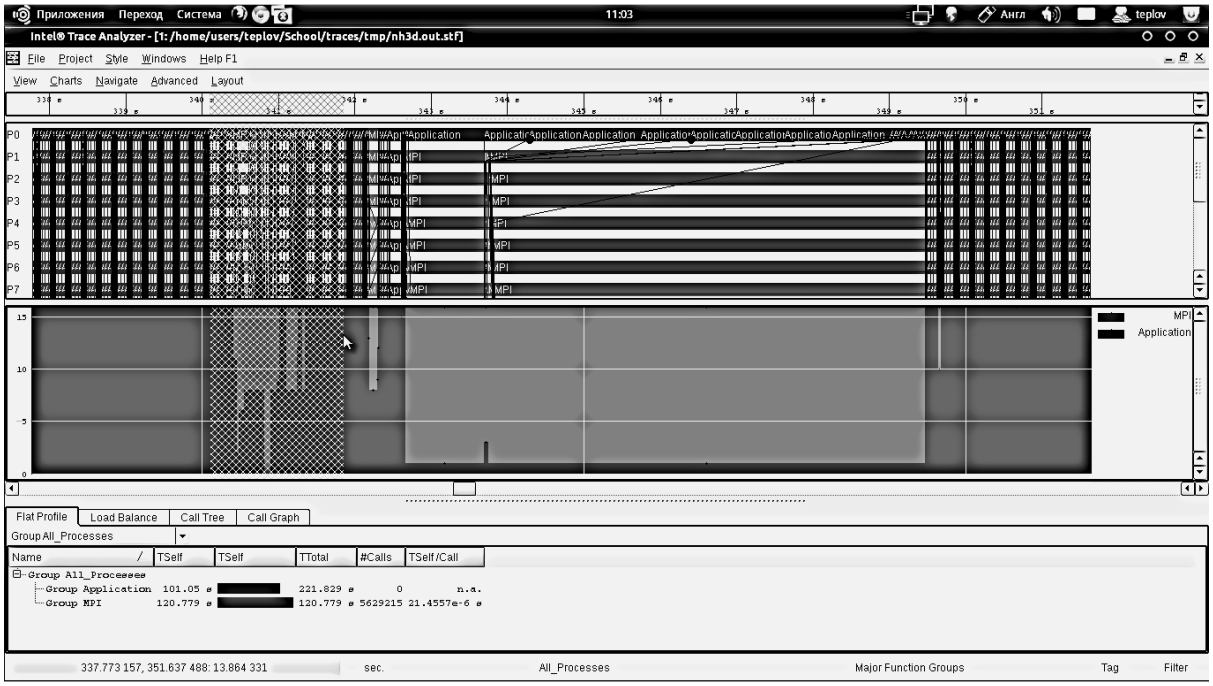

Рис 1. Пример участка трассы работы приложения, на котором работу выполняет только один процесс

Для выявления потенциальных узких мест удобно пользоваться количественной диаграммой (на рисунках находится под диаграммой событий). На ней отображается число процессов, участвующих в выполнении каждой операции. Идеальная картина – все процессы заняты одной и той же деятельностью. Если наблюдается «лестница», то этот участок не оптимален. Таких мест в модели NH3D достаточно много.

На втором этапе анализа было определено расположение процедур, замедляющиеся с увеличением числа процессоров. Для выяснения причин отсутствия ускорения проводилось инструментирование исходного кода программы с помощью API, предоставляемого ITAC.

В коде были выделены два класса функций: функции инициализации и функции шага по времени. Для такого разделения необходимо передавать параметр идентификатор вызывающего класса функций.

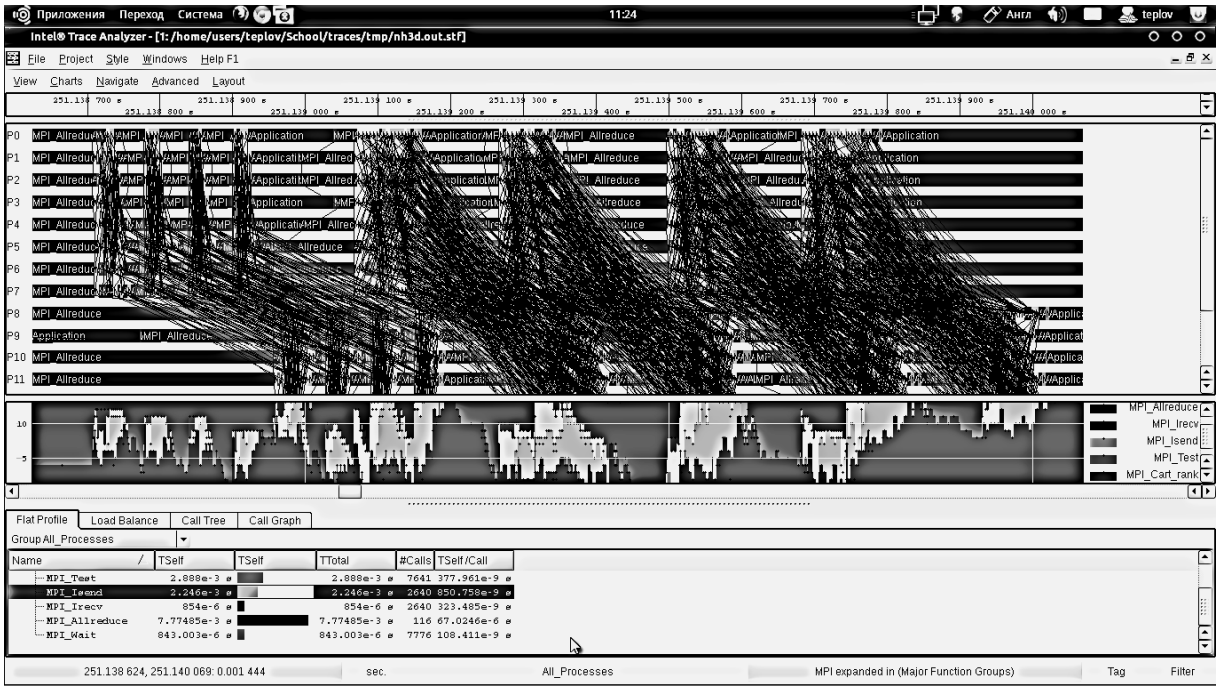

Рис 2. Пример участка трассы приложения,
на котором работа процессов разбалансирована

Сбор трасс проводился с использованием алгоритма усреднения данных на границе области SPONGE и без него. На рис. 3 показана трасса работы процедуры MOMENT_MPI на 25 процессорах с использованием алгоритма SPONGE, а на рис. 4 – без его использования.

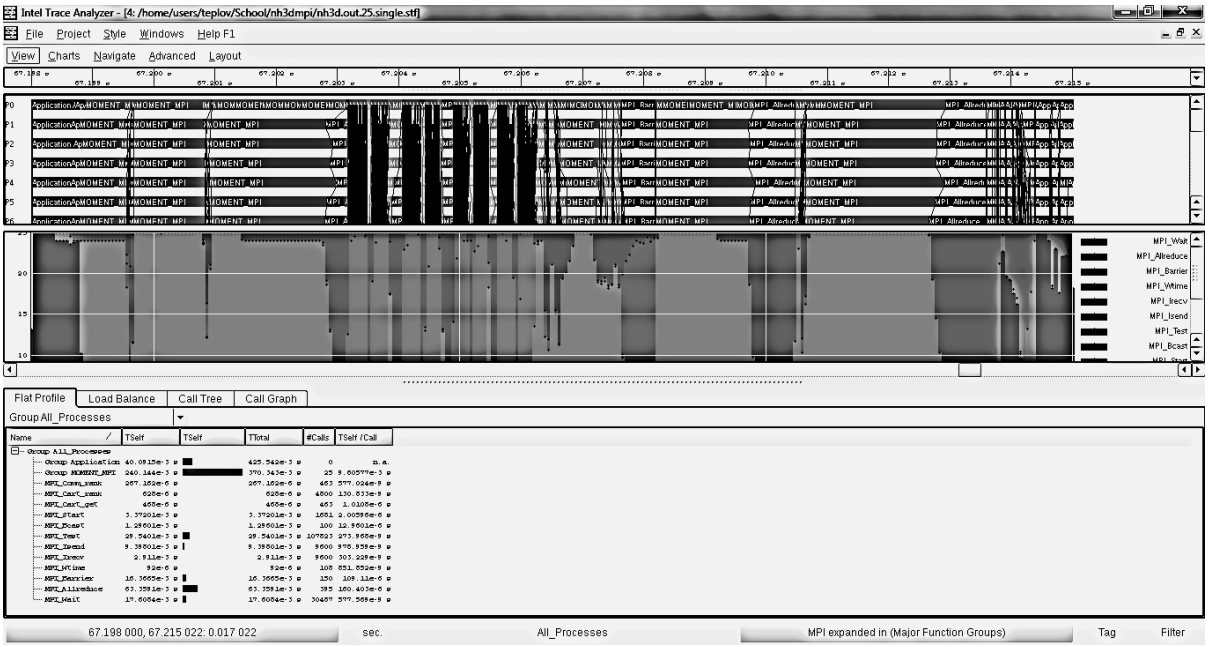

Рис 3. Участок трассы программы,
на котором процедура MOMENT_MPI использует алгоритм SPONGE

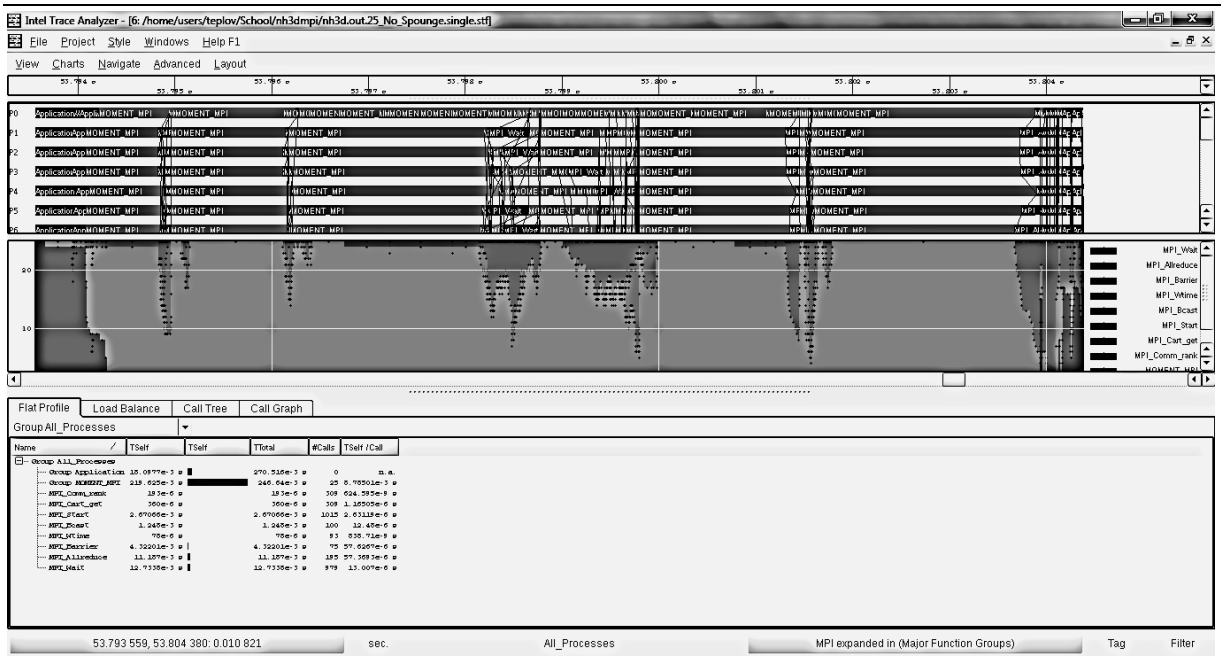

Рис 4. Участок трассы программы,
на котором процедура MOMENT_MPI не использует алгоритм SPONGE

При сравнении двух трасс легко заметить отличия в характере профиля работы процедуры и влияние процедуры SPONGE_MPI. Таким образом, можно заключить, что SPONGE_MPI является узким местом, сильно замедляющим работу приложения. На следующем этапе работы проводилось более глубокое инструментирование шагов модели по времени. На трассу выводились данные обо всех процедурах, которые используют вызовы процедур MPI. Основную сложность при инструментировании вызвали вложенные процедуры, которые используются в нескольких процедурах. Необходимо корректно учитывать отображение на трассе времени выполнения кода вложенных процедур (run time) и кода самой процедуры (self time) всех процедур. Если основу времени выполнения процедуры занимает выполнение вложенной процедуры, то больше внимания при оптимизации необходимо уделить вложенной процедуре.

В ходе такого типа анализа была выявлена процедура UPUVT_MPI, которая вызывается из многих процедур, и потому занимает существенную часть времени шага модели. Работа процедуры была проанализирована на основе собранных трасс и оптимизирована, исходя из результатов исследования.

7. Данные о масштабируемости приложения, полученные с помощью ITAC

Целью работы являлся анализ масштабируемости приложения, и потому данные, предоставляемые инструментами исследования параллельных приложений, должны быть рассмотрены в совокупности всех проведенных экспериментов. Представление о масштабируемости приложения возможно получить только после профилирования всех процедур и получения сводной статистики серии экспериментов.

Анализ статистических данных о метриках эффективности параллельного приложения, вне зависимости от выбранной метрики и способа сбора данных, позволяет сделать выводы о наличии проблем масштабируемости приложения. Для данной задачи использование ITAC позволило быстро собрать статистические данные экспериментов.

После определения наличия проблемы для выявления её места в коде использовалась количественная диаграмма. По ней определялись участки кода, в которых происходит разбалансировка работы.

Более детальный поиск проблем с помощью инструментирования исходного кода позволил оценить вклад каждой процедуры в масштабируемость всего приложения и выявить наиболее критичные. После этого статистика о работе выделенного участка собиралась в сводную картину масштабируемости всего приложения и его частей.

После локализации узких мест необходимо сделать выводы о причинах плохой масштабируемости в найденных участках. Узкие места более детально исследовались на диаграммах событий. Изучая в совокупности диаграммы событий (рис 5.A) и качественные диаграммы (рис 5.B), делались выводы о том, какие вызовы MPI приводят к разбалансировкам, а также каков их вклад. Для анализа данных серии экспериментов возможностей анализатора по сравнению различных трасс было недостаточно. Сравнение двух трасс не давало полноценной картины масштабируемости. Однако, сравнивая минимальные и максимальные конфигурации по числу использованных процессоров, можно получить представление о том, как сильно меняется профиль работы.

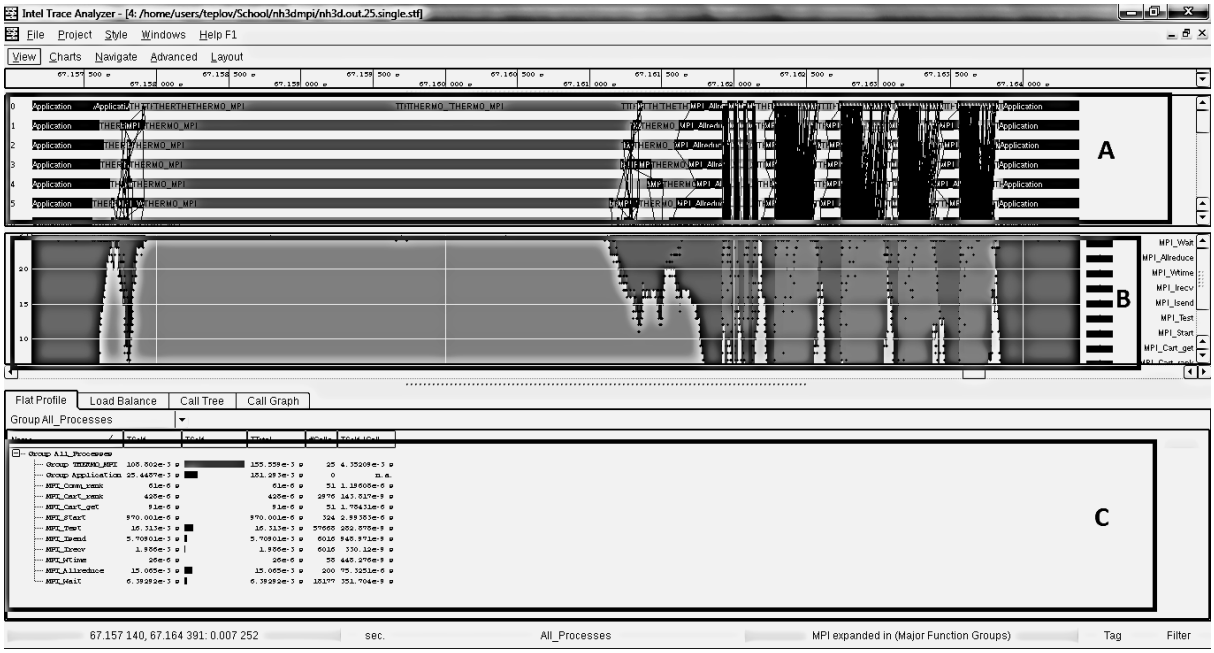

Рис 5. A – диаграмма событий. B – качественная диаграмма.
C – профиль отображенного сегмента трассы

Другой проблемой стал поиск узких мест при больших масштабах вычислительной системы. Общее число отображаемых событий можно сократить, применяя фильтрацию. При таких масштабах визуализацию диаграммы событий невозможно уместить на одном экране, поэтому анализ затруднен в силу отсутствия автоматического определения проблем эффективности в данном инструменте.

Инструмент не позволил выявить проблемы распределения данных. Анализатор разбалансировки приложения предоставляет также данные об объемах и количестве коммуникационных сообщений.

Заключение

Делая выводы об использовании ITAC в целях исследования масштабируемости приложения, нужно отметить, что данный инструмент предоставляет достаточно данных для того, чтобы сделать однозначные выводы о существовании проблем масштабируемости. Возможностей инструмента достаточно для локализации проблем, а также получения представления о масштабируемости отдельных частей приложения. При детальном анализе узких мест с помощью ITAC можно получить представление о характере проблем исходного кода. Инструмент может анализировать трассы работы на больших конфигурациях вычислительной системы. И потому, несмотря на сложность анализа трасс с большим количеством процессоров, ITAC предоставляет достаточно информации для того, чтобы определить наличие проблем масштабируемости, локализовать узкие места программы и получить представление о характере проблемы.

Работа выполняется при поддержке гранта РФФИ 10-07-00586-а.

Литература

1. Grama, A. Introduction to Parallel Computing. (2nd Edition) / A. Gupta, G. Karypis, V. Kumar. – Pearson, 2003.
2. Иванников, В.П. Оценка динамических характеристик параллельной программы на модели / В.П. Иванников, С.С. Гайсарян, В.А. Падарян // Программирование. – 2006. – № 4.
3. Alabdulkareem, M. Scalability Analysis of Large Codes Using Factorial Designs / M. Alabdulkareem, S. Lakshmivarahan, S.K. Dhall // Parallel Computing. – 2001. – Vol. 27, Issue 9. – P. 1145–1171.
4. Muller-Wichards, D. Scalability of Algorithms: An Analytic Approach / D. Muller-Wichards, W. Ronsch // Parallel Computing. – 1995. – Vol. 21, Issue 6. – P. 937–952.
5. Степаненко, В.М. Численное моделирование мезомасштабной динамики атмосферы и переноса примеси над гидрологически неоднородной территорией / В.М. Степаненко, Д.Н. Микушин // Вычислительные технологии. – 2008. – Т. 13, специальный выпуск 3. – С. 104–110.

Антонов Александр Сергеевич, к.ф.-м.н., старший научный сотрудник, НИВЦ МГУ имени М.В. Ломоносова, asa@parallel.ru.

Теплов Алексей Михайлович, младший научный сотрудник, НИВЦ МГУ имени М.В. Ломоносова, alex-teplov@yandex.ru.

APPLICATION SCALABILITY STUDY USING TOOLS TO ANALYZE THE PARALLEL APPLICATIONS ON THE EXAMPLE OF THE ATMOSPHERE MODEL NH3D

A.S. Antonov, RCC MSU (Moscow, Russian Federation),
A.M. Teplov, RCC MSU (Moscow, Russian Federation)

The article describes an approach to the scalability study using applications analyzing tools. This approach uses application profiling and tracing of the bottle necks for scalability analysis. There is a brief review of parallel program effectiveness marks, approaches to scalability study and performance analysis tools for parallel applications. This article performs detailed description of analyzing methodology and detailed review of parallel application that was analyzed using this methodology. Also in this article was performed the analysis results of described parallel application using profiling and tracing tools. For the analysis the authors used Intel Trace Analyzer and Collector as an example of tracing tool. In conclusion the authors admitted high level of analyzing features of this tool for scalability analysis purposes.

Keywords: scalability, effectiveness analysis, tracing, profiling, parallel application analysis tools.

References

1. Grama A., Gupta A., Karypis G., Kumar V. Introduction to Parallel Computing. (2nd Edition), Pearson, 2003.
2. Ivannikov V.P., Gaysaryan S.S., Padaryan V.A. Evaluation of Dynamic Characteristics of a Parallel Program on a Model. Programing and Computer Software. 2006. No. 4.
3. Alabdulkareem M., Lakshmivarahan S., Dhall S.K. Scalability analysis of large codes using factorial designs. Parallel Computing. 2001. Vol. 27. Issue 9. P. 1145–1171.
4. Muller-Wichards D., Ronsch W. Scalability of Algorithms: An Analytic Approach. Parallel Computing. 1995. Vol. 21. Issue 6. P. 937–952.
5. Stepanenko V.M., Mikushin D.N.. Numerical modeling of mesoscale atmospheric dynamics and pollutant transport over hydrologically diversified area // Computational technologies. 2008. Vol. 13, special issue 3. P. 104–110.

Поступила в редакцию 10 января 2013 г.

УДК 004.94

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ЗАДАЧАХ ФИЗИКИ ВЫСОКИХ ЭНЕРГИЙ ДЛЯ ПРОЕКТА NICA/MPD[1]

А.Е. Басалаев, А.С. Бондаренко, С.П. Мерц, С.А. Немнюгин, А.Н. Тимофеев

Представлен подход к оптимизации вычислений, основанный на надстройке PROOF для пакета научных вычислений ROOT. Метод успешно применялся в задачах, возникающих в проекте NICA/MPD, таких как моделирование ridge-эффекта и вычисление радиационных длин детектора MPD. Получены зависимости ускорения от различных параметров.

Ключевые слова: NICA/MPD, MpdRoot, Ridge-effect, высокопроизводительные вычисления, PROOF, UrQMD, HADGEN

Введение

Любой современный эксперимент в области физики элементарных частиц требует тщательного моделирования. На этом этапе происходит апробация применяемых алгоритмов, сравнение используемых моделей, тестирование и оптимизация написанного программного кода. Под оптимизацией здесь понимается как алгоритмическое усовершенствование кода, так и внедрение высокопроизводительных и параллельных технологий.

В рамках научной программы по изучению горячей и плотной барионной материи в ОИЯИ (Объединенный Институт Ядерных Исследований, Дубна) реализуется проект по созданию нового ускорительного комплекса с встречными пучками NICA (Nuclotron-based Ion Collider fAcility) [1, 2]. Новый ускорительный комплекс позволит исследовать взаимодействия тяжелых ионов в широком диапазоне атомных масс: от легких ядер до ядер золота при энергии 3–11 ГэВ/нуклон в системе центра масс и светимости 10^{27}см$^{-2}$с$^{-1}$ при частоте 6 кГц.

На коллайдере предусмотрены две точки пересечения пучков, в одной из которых будет располагаться экспериментальная установка MPD (Multi-Purpose Detector, многоцелевой детектор), предназначенная для исследований столкновений тяжелых ионов [3].

Детектор MPD состоит из цилиндрической и двух торцевых частей (рис. 1). Основным детектором для регистрации треков заряженных частиц и их идентификации в центральной области MPD является время-проекционная камера TPC (Time Projection Chamber).

В данной работе рассмотрены задачи оптимизации процесса вычисления, возникающие при моделировании работы установки NICA/MPD.

Для генерации событий, моделирования работы детектора и реконструкции событий на установке NICA/MPD разрабатывается программный комплекс MpdRoot [4], основанный на пакете для научных расчетов ROOT [5]. Данный комплекс предоставляет множество инструментов как для работы с данными, так и для моделирования в целом, и допускает подключение внешних библиотек для расширения функциональных возможностей.

[1]Статья рекомендована к публикации программным комитетом международной суперкомпьютерной конференции «Научный сервис в сети интернет. Поиск новых решений — 2012»

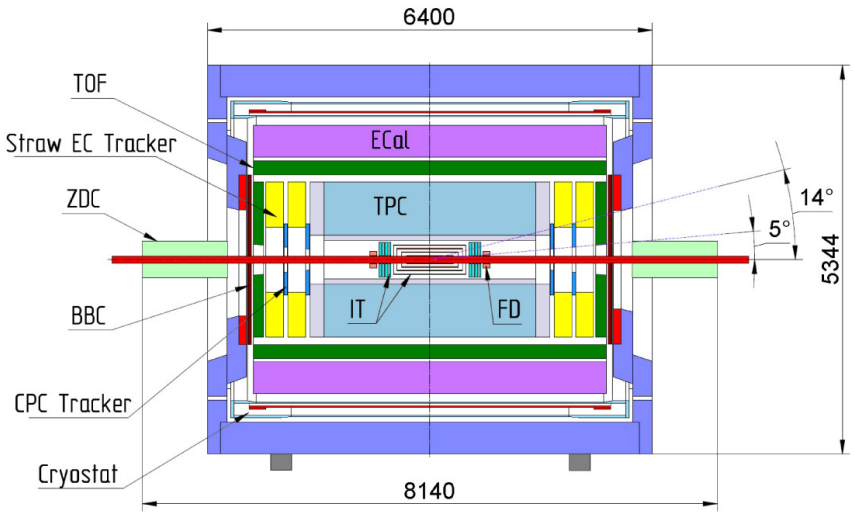

Рис. 1. Схема многоцелевого детектора MPD

1. Использование облачных вычислений

В настоящее время ресурсный вычислительный центр Санкт-Петербургского государственного университета располагает несколькими вычислительными системами, одна из которых построена на принципах виртуализации. Система состоит из двух корзин blade-серверов HP BladeSystem c7000, включающих в себя по 16 модулей BL460G7 — 2 процессора Intel Xeon X5670 и 96Гб оперативной памяти каждый. Для хранения данных используются две системы HP StorageWorks P4500 G2 емкостью 240 ТБ каждая. Характеристики системы: 32 сервера, 64 процессора, 384 ядра, 3 ТБайт ОЗУ, 480 ТБайт дискового пространства, пиковая производительность 4,5 Тфлопс. Вычислительная система находится под управлением VMWare vSphere [6].

Для работы из общей системы был выделен массив из 10 виртуальных серверов, каждый из которых оснащен четырехъядерным процессором Intel Xeon Processor X5670 и 10 Гб оперативной памяти. Используемые средства виртуализации позволяют реализовать динамическое распределение серверных ресурсов, обеспечивая предоставление нужного ресурса нужному приложению в соответствии с приоритетами. Среды исполнения приложений могут увеличиваться и уменьшаться при необходимости: есть возможность «горячего» добавления ЦП и памяти к виртуальным машинам при необходимости и без прерывания, возможность «горячего» расширения виртуальных дисков обеспечивает добавление пространства для хранения данных к работающим виртуальным машинам. Такая возможность динамического выделения ресурсов позволяет оптимизировать нагрузки и энергопотребление, что особенно важно для высоконагруженных научных вычислительных систем.

2. Разработка интерфейса между генераторами событий и MpdRoot

Модель ядерных столкновений UrQMD (Ultrarelativistic Quantum Molecular Dynamics model, ультрарелятивистская квантово-молекулярная динамическая модель, [7]) и HADGEN

(HADron GENerator, адронный генератор) являются зарекомендовавшими себя во многих экспериментах инструментами для моделирования столкновений тяжелых ядер, но работа с ними не всегда удобна. Данные генераторы работают в одном потоке, а время, необходимое для генерации большого количества событий, весьма значительно; помимо этого, выходные данные сохраняются в виде файлов, что увеличивает затраты на операции чтения/записи, и сами файлы могут занимать десятки гигабайт. Модель ядерных столкновений UrQMD фактически является стандартом в области физики высоки энергий; выбор программного генератора HADGEN обусловлен его высокой точностью описания процессов, происходящих при энергиях NICA (в диапазоне от 3 до 11 ГэВ/нуклон), и тем, что он является составной частью пакета SHIELD [8], планируемого к использованию для расчета радиационной защиты ускорителя и многих его элементов.

В данной работе на примере двух генераторов UrQMD и HADGEN предлагается простой и достаточно универсальный способ ускорить моделирование и обработку данных за счет использования интерфейса подключения этих библиотек к программному комплексу MpdRoot и использования его расширения PROOF (Parallel ROOT facility, [9]).

Оригинальные исходные коды генераторов UrQMD и HADGEN написаны на языке FORTRAN; хранение данных осуществляется в глобальных структурах (common-blocks). Использование глобальных переменных имеет два основных недостатка: 1) снижение надежности программ, их модульности и противоречие принципам инкапсуляции, 2) отсутствие возможности реализовать параллелизм задач такими средствами языка, как POSIX threads или аналогичными инструментами.

В качестве первого шага оптимизации были разработаны интерфейсы на языке C, реализация которых обеспечивает безопасный доступ к данным (рис. 2). Исходные коды генераторов вместе с интерфейсами собираются в динамические библиотеки.

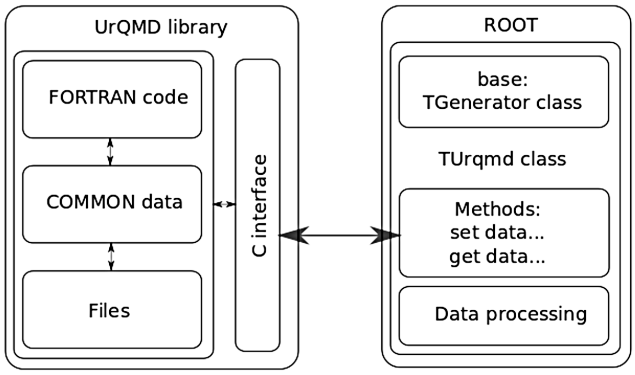

Рис. 2. Архитектура враппера для модели ядерных столкновений UrQMD

Вторым шагом является создание в рамках MpdRoot классов-оболочек (wrappers) на языке C++, взаимодействующих с интерфейсами библиотек. Данные классы содержат методы, необходимые для работы с генератором, введения начальных параметров моделирования и получения выходных данных в формате, удобном для дальнейшего использования.

Для достижения максимальной производительности был использован инструмент PROOF — расширение пакета ROOT, позволяющее реализовать интерактивную, параллельную обработку больших массивов данных или, в общем случае, добиться параллельности выполнения кода на уровне задач.

PROOF-кластер строится по стандартной схеме master-slave (рис. 3). Благодаря многоуровневой архитектуре, позволяющей создать иерархию из master и submaster узлов, такой подход может быть легко адаптирован к широкому диапазону виртуальных кластеров, географически распределённых между доменами и гетерогенными машинами (GRID). Клиентом системы является пользователь, который хочет использовать ресурсы сайта, чтобы выполнить свою задачу. Master — это точка входа к вычислительному средству: он разбирает запросы клиента, распределяет работу между ресурсами, собирает и соединяет результаты. Координация работы отдельных серверов достигается за счет использования специального протокола передачи данных PROOF.

Пользователь, работая в сессии программы ROOT, может запускать процессы, которые связываются с PROOF-кластером и подают запросы на обработку заданий. Получив запрос на обработку задания, на мастере и на рабочих узлах для каждой сессии пользователя стартует специальное ROOT-приложение - proofserv. Процесс, исполняющийся на мастере, координирует работу между рабочими узлами и объединяет результаты в единое целое. На рабочих узлах процесс proofserv делает непосредственно вычислительную работу, обрабатывая отдельные задания.

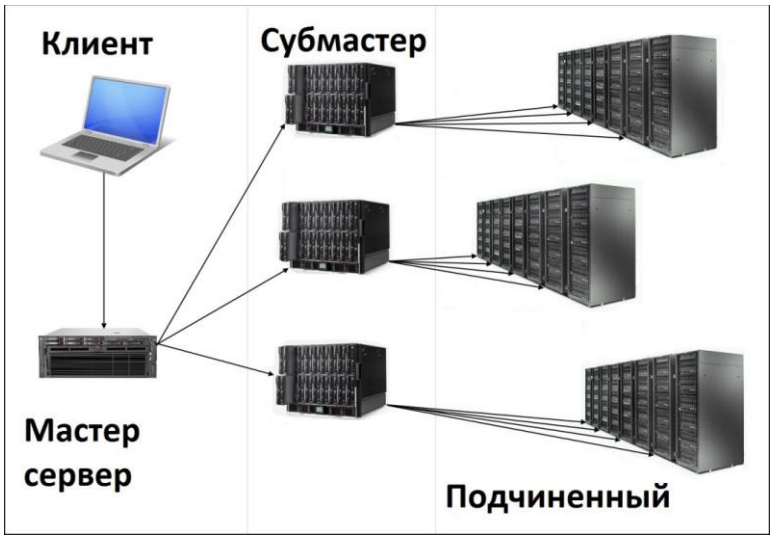

Рис. 3. Схема кластера PROOF

Задания для обработки на PROOF-кластере пишутся особым образом; для этого объявляется класс C++, наследуемый от встроенного в ROOT класса TSelector. В данном классе обязательно должен быть переопределен набор методов, реализующий функционал вычислительного узла (рис. 4).

Для работы с PROOF-кластером создается экземпляр класса TProof (рис. 5). Функция void Process(..) служит для запуска задания в параллельном режиме с заданным количеством итераций nIterations.

В ходе решения задач проекта NICA появилась потребность в моделировании Ridge-эффекта [10] для подробного рассмотрения структуры события столкновения тяжелых ядер. Природа данного эффекта на данный момент не объяснена, существует несколько теорий. Эффект состоит в наличии корреляций между частицами, получающимися при столкновении тяжелых ядер. Имеется корреляция по азимутальному углу даже “далеких” друг

Рис. 4. Иерархия вызовов обязательных методов при работе с PROOF

```
TProof * p = TProof::Open("proof.cluster.address");

p->SetParallel(nThreads);

p->Process("ClassImplementation.C", nIterations);
```

Рис. 5. Пример работы с PROOF-кластером

от друга частиц, показывающему отклонение частиц в сторону. Для наблюдения Ridge-эффекта предъявляются некоторые требования по отбору частиц: величина их поперечного импульса должна лежать в определенных пределах. Задача моделирования данного эффекта во многом обусловлена необходимостью объяснить, почему только частицы с определенным спектром импульсов имеют подобные корреляции. Для этого требуется моделирование с большим объемом статистики (порядка миллионов событий), поэтому был использован кластер виртуальных машин PROOF. Благодаря тому, что написание класса оболочки намного проще, чем попытки внедрить параллелизм в код самого генератора, и наличию легко масштабируемой вычислительной системы, удалось добиться высокой производительности с малыми затратами.

Ускорение, как видно из рис. 6, ниже теоретического предела Амдала. В данном примере, каждый поток моделировал 100 столкновений тяжелых ядер золота. При увеличении нагрузки на поток наблюдается увеличение производительности, так как потери на пересылку данных и накладные расходы на создание и мониторинг потоков, становятся малыми по сравнению с временем вычислений.

3. Вычисление радиационных длин для детектора MPD

Одной из актуальных задач на стадии проектирования эксперимента NICA/MPD является исследование возможности помещения дополнительных трековых детекторов за торцевыми стенками TPC (см. рис. 1), на которых размещена электроника для съема информации. Важной является оценка влияния аппаратуры на торцевых стенках TPC на пролета-

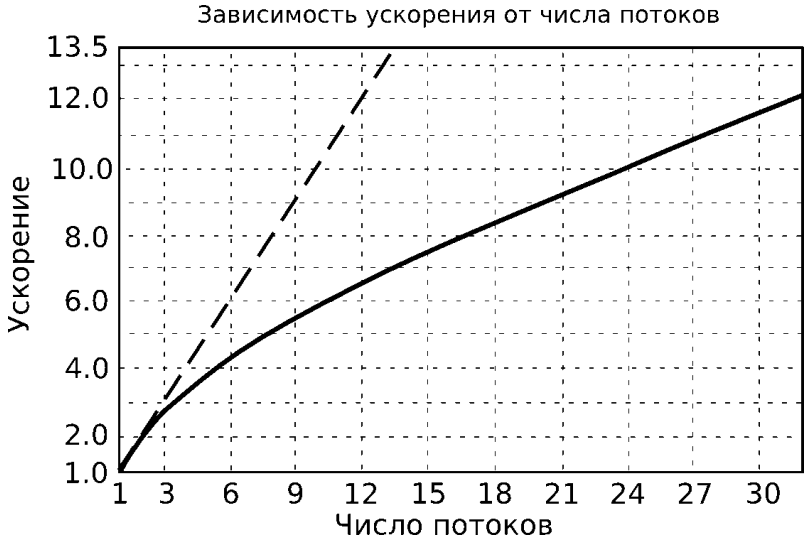

Рис. 6. Зависимость ускорения генерации событий в UrQMD от числа потоков при использовании многоядерной архитектуры PROOF

ющие сквозь них частицы. Это влияние носит случайный характер и скорректировать его невозможно. Характеристикой материала, показывающей, насколько велико это влияние, является радиационная длина - средняя толщина вещества, на которой энергия электрона уменьшается в e раз, умноженная на плотность вещества. Её необходимо минимизировать.

Для выполнения этой задачи необходимо получить распределение радиационных длин в детекторе и вычислить интегральную радиационную длину для TPC. Точность расчетов зависит от плотности покрытия точками области сканирования. Поэтому одной из задач в данной работе было не просто вычислить радиационные длины для MPD, но сделать это с высоким разрешением.

Данная задача была решена на виртуальном кластере в среде MpdRoot. Для ускорения расчетов применялось распараллеливание с помощью PROOF.

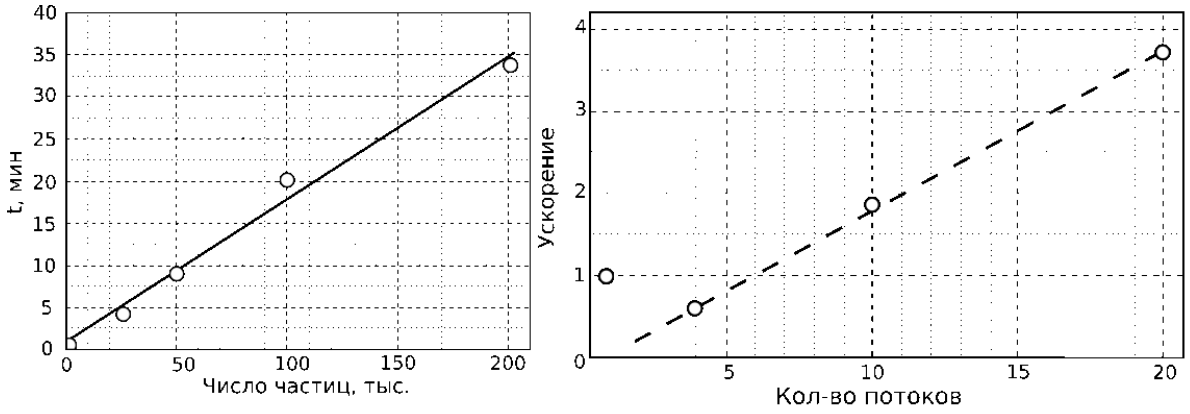

Рис. 7. Анализ производительности кода для расчета радиационных длин

Использование PROOF при проведении расчета в одном потоке сильно снижает производительность, как видно на рис. 7. Однако при использовании большого количества потоков наблюдается выигрыш в скорости, при этом рост производительности близок к линейно-

му. Безусловно, конкретная задача еще нуждается в дополнительной оптимизации, однако преимущество PROOF очевидно.

На рис. 8 представлен результат моделирования — картина детектора MPD в радиационных длинах.

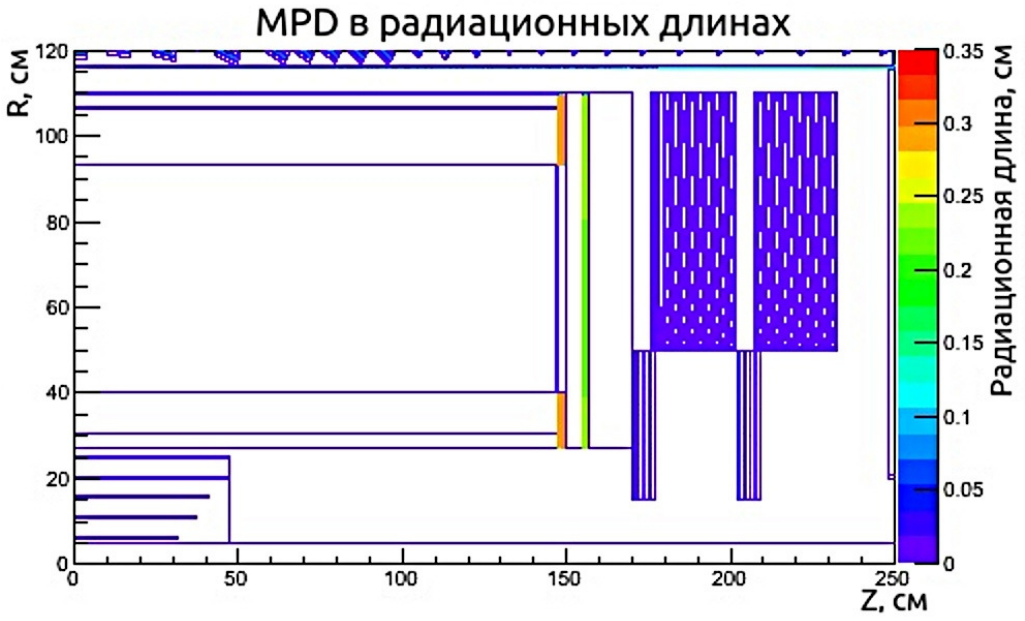

Рис. 8. Представление MPD в радиационных длинах

Авторы выражают благодарность ресурсному центру «Вычислительный центр СПбГУ» за предоставленные вычислительные системы, на которых были произведены расчеты.

Литература

1. Ускорительно-накопительный комплекс NICA. Технический проект / под общ. ред. И.Н. Мешкова и А.О, Сидорина – Дубна: ОИЯИ. 2009. 72 с.
2. Ускорительно-накопительный комплекс NICA — база фундаментальных исследований и инновационных разработок / под общ. ред. В.Д. Кекелидзе; сост.: А.Д. Коваленко и др. – Дубна: ОИЯИ, 2011. – 32 с.
3. Abraamyan, Kh.U. et al. The MPD detector at the NICA heavy-ion collider at JINR // Nuclear Instruments and Methods in Physics Research. – 2011. – A628. – P. 99 – 102.
4. Simulation and Analysis Framework for NICA/MPD Detectors. URL: `http://mpd.jinr.ru` (дата обращения: 22.12.2012).
5. ROOT – A Data Analysis Framework. URL: `http://root.cern.ch` (дата обращения: 21.12.2012).
6. Ресурсный вычислительный центр СПбГУ. URL: `http://ptc.spbu.ru/hpc` (дата обращения: 22.12.2012).

7. Bleicher, M. et al. Relativistic Hadron-Hadron Collisions in the Ultra-Relativistic Quantum Molecular Dynamics Model // J. Phys. G: Nucl. Part. Phys. – 1999. – Vol. 25. – P. 1859–1896.
8. Dementyev, A.V. SHIELD, a Monte Carlo Hadron Transport Code / A.B. Dementyev, N.M. Sobolevsky // Institute of Nuclear Research of Russian Academy of Science – Moscow.
9. Brun, R. et al. Parallel interactive data analysis with PROOF // Nuclear Instruments and Methods in Physics Research. – 2006. – A559. – P. 13–16.
10. STAR Collaboration. Correlation Structures from Soft and Semi-hard Components in pp Collisions at s = 200 GeV. // Acta Phys. Polon. – 2005. – B36. – P. 353.

Артем Евгеньевич Басалаев, студент, кафедра вычислительной физики, Санкт-Петербургский государственный университет (г. Санкт-Петербург, Российская Федерация), artem.basalaev@gmail.com

Артем Сергеевич Бондаренко, аспирант, кафедра вычислительной физики, Санкт-Петербургский государственный университет (г. Санкт-Петербург, Российская Федерация), tema.bond@gmail.com

Сергей Павлович Мерц, младший научный сотрудник, лаборатория физики высоких энергий, Объединенный институт ядерных исследований (г. Дубна, Российская Федерация), Sergey.Merts@gmail.com

Сергей Андреевич Немнюгин, кандидат физико-математических наук, доцент, кафедра вычислительной физики, Санкт-Петербургский государственный университет (г. Санкт-Петербург, Российская Федерация), snemnyugin@mail.ru

Александр Николаевич Тимофеев, аспирант, кафедра вычислительной физики, Санкт-Петербургский государственный университет (г. Санкт-Петербург, Российская Федерация), antimofeew@gmail.com

---

USING OF THE HIGH-PERFORMANCE COMPUTING TECHNOLOGIES IN HIGH ENERGY PYSICS TASKS FOR NICA/MPD

A.E. Basalaev Saint-Petersburg State University (Saint-Petersburg, Russian Federation),
A.S. Bondarenko Saint-Petersburg State University (Saint-Petersburg, Russian Federation),
S.P. Merts Joint Institute for Nuclear Research (Dubna, Russian Federation),
S.A. Nemnugin Saint-Petersburg State University (Saint-Petersburg, Russian Federation),
A.N. Timofeev Saint-Petersburg State University (Saint-Petersburg, Russian Federation)

The article deals with the approach to optimisation of calculation by PROOF. The method was used successfuly in the tasks of experiment NICA/MPD. For example ridge-effect modeling and

radiation lenght calculation for detector MPD. Acceleration dependencies of different parameters are present.

Keywords: NICA/MPD, MpdRoot, Ridge-effect, high-performance computing, PROOF, UrQMD, HADGEN

References

1. Meshkova I.N., Sidorina A.O. Uskoritel'no-nakopitel'nyj kompleks NICA. Tehnicheskij proekt [Superconducting accelerator complex NICA. Technical project]. Dubna: JINR. 2009. 72 p.
2. Kekelidze V.D. et al. Uskoritel'no-nakopitel'nyj kompleks NICA — baza fundamental'nyh issledovanij i innovacionnyh razrabotok [Superconducting accelerator complex NICA — the basis for fundamental research and innovations]. Dubna: JINR, 2011. 32 p.
3. Abraamyan Kh.U. et al. The MPD detector at the NICA heavy-ion collider at JINR. Nuclear Instruments and Methods in Physics Research. 2011. A628. P. 99 – 102.
4. Simulation and Analysis Framework for NICA/MPD Detectors. URL: `http://mpd.jinr.ru`.
5. ROOT — A Data Analysis Framework. `http://root.cern.ch`.
6. Resursnyj vychislitel'nyj centr SPbGU [SPBU computing center] URL: `http://ptc.spbu.ru/hpc`.
7. Bleicher M. et al. Relativistic Hadron-Hadron Collisions in the Ultra-Relativistic Quantum Molecular Dynamics Model. J. Phys. G: Nucl. Part. Phys.1999. Vol. 25. P. 1859 – 1896.
8. Dementyev A.V., Sobolevsky N.M. SHIELD, a Monte Carlo Hadron Transport Code. Institute of Nuclear Research of Russian Academy of Science. Moscow.
9. Brun R. et al. Parallel interactive data analysis with PROOF. Nuclear Instruments and Methods in Physics Research. 2006. A559. P. 13–16.
10. STAR Collaboration. Correlation Structures from Soft and Semi-hard Components in pp Collisions at s = 200 GeV. Acta Phys. Polon. 2005. B36. P. 353.

Поступила в редакцию 23 октября 2012 г.

УДК 004.65:548.0 : 549.02

WWW-MINCRYST: ИНТЕРНЕТ-ОРИЕНТИРОВАННАЯ ИНФОРМАЦИОННО-ВЫЧИСЛИТЕЛЬНАЯ СИСТЕМА ПО КРИСТАЛЛОГРАФИИ И КРИСТАЛЛОХИМИИ МИНЕРАЛОВ[1]

Д.А. Варламов, Т.Н. Докина, Н.А. Дрожжина, О.Л. Самохвалова

Описывается интернет-ориентированная информационно-вычислительная система (ИВС) WWW-MINCRYST, предназначенная для работы с кристаллическими структурами минералов, их синтетических аналогов и элементов. Основными компонентами ИВС являются собственно база данных (более 8000 записей для 3000 уникальных фаз), снабженная комплексом средств поиска и выбора информации, средствами мультимедийного представления информации (полиэдрические и шаровые интерактивные структуры, спектры и т.п.), возможностями обработки спектральной информации, в том числе с использованием пользовательских данных. ИВС размещена по адресу http://mincryst.iem.ac.ru и доступна пользователям без ограничений.

Ключевые слова: интернет-ориентированные базы данных, кристаллохимия, кристаллические структуры, минералы, PHP, MySQL.

Введение

WWW-MINCRYST (http://mincryst.iem.ac.ru) в качестве Интернет-ориентированной базы данных возник достаточно давно и стал одним из пионерских для РФ научных интерактивных Интернет-ресурсов (первый полностью работоспособный вариант WWW интерфейса был представлен пользователям уже в декабре 1997 года [1]). Ресурс призван обеспечить интерактивный доступ Интернет-пользователям к накапливавшимся с 1985 года данным по кристаллическим структурам (прежде всего литературным, а также авторским аналитическим) и многочисленным разработкам авторов проекта по обработке этих данных и анализу. Инициатором и основным идеологом работ стал заведующий группой рентгеноспектрального анализа выдающийся кристаллограф А.В.Чичагов (увы, ушедший от нас в 2010 году). Методология создания и технические детали текущей реализации информационно-вычислительной системы (ИВС) уже освещались ранее [2, 3] и здесь описана кратко, также освещены текущее состояние ИВС и перспективы ее развития.

В настоящее время ИВС WWW-MINCRYST (по оценкам пользователей и составителей каталогов информационных ссылок) входит в первые ряды рентгеноструктурных баз данных, связанных с изучением минерального вещества.

Созданная ранее и развиваемая ИВС с ее инструментарием во многом решает важную задачу информационного обеспечения исследований во всех областях науки, оперирующих с кристаллическим веществом (минералогия, кристаллография, физика твердого тела и т.п.), обеспечивая как поддержку многочисленных исследователей минерального

[1] Статья рекомендована к публикации программным комитетом Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: поиск новых решений – 2012».

вещества и синтетических аналогов (включая студентов и аспирантов), так и давая дополнительный толчок в изучении кристаллографических данных.

WWW-MINCRYST зарегистрирован в Государственном Регистре Баз Данных (НТЦ «Информрегистр») под номером 0229805169 (Регистрационное Свидетельство № 4873 от 11.02.99).

1. Строение и идеология ИВС WWW-MINCRYST

В основе WWW-MINCRYST лежит ряд принципиальных «идеологических» моментов, реализация которых уже обеспечила ему достаточное признание мировой кристаллографической и минералогической общественности.

Синтез информации о кристаллической фазе, рассматриваемой как "монокристалл" и/или как "поликристалл" с заменой экспериментальных поликристалл-стандартов расчетными – стержневая идея WWW-MINCRYST, которая и делает его универсальной, комплексной и оригинальной разработкой.

В настоящее время информационный фонд WWW-MINCRYST насчитывает чуть более 8000 записей для более чем 3000 уникальных фаз (минералов, их аналогов, элементов), т.е. большинства минералов (из 4800), кристаллические структуры которых расшифрованы к настоящему времени. Помимо природных объектов, в базе данных представлены синтетические минералы – их структурные аналоги, отличающиеся по составу (например, с заменой одного из катионов), и неорганические соединения (силикаты, фосфаты, бораты и т.п.), близкие по свойствам к природным веществам. Информационный фонд содержит данные структурных работ из более 120 иностранных и отечественных журналов за период от 1930-х до 2012 года. Ежегодный прирост новых или заново переопределенных кристаллических структур минералов и их аналогов достаточно значителен, чтобы требовалась постоянная актуализация информационного фонда (в среднем, до 300-400 структур в год). В последние годы основной акцент сделан на новые минералы, а также на кристаллические структуры, описанные в советских и российских журналах (которые в последние десятилетия были менее доступны web-пользователю, нежели зарубежные). Суммарный объем базы составляет около 500 Мб.

Технологически WWW-MINCRYST как сервис реализован на «классической» связке Linux-Apache-Mysql-PHP (LAMP) с использованием JavaScript и (для интерактивных аплетов) Java (на базе Sun/Oracle JDK), реализованной на сервере баз данных ИЭМ РАН (http://database.iem.ac.ru). WWW-MINCRYST был (в 1997 году) одной из первых интернет-ориентированных баз данных в России, использовавшим данную технологию, причем за 14 лет она нисколько не потеряла свою технологичность на стороне сервера. ИВС является двуязычной, включая содержание и языки интерфейса: русский и английский. В систему входят следующие компоненты: (а) собственно база данных; (б) комплексные поисковые интерфейсы, (в) мультимедийные интерактивные формы представления структурной и спектральной информации (через Java-апплеты); (г) классификационные схемы; (д) системы динамически формируемых ссылок на внешние информационные ресурсы; (е) WWW-ориентированный инструментарий разработчика; (ж) прикладные программы по обработке кристаллографической информации (WWW-XrayPol и WWW-MixiPol).

Основной источник информации для Базы данных по кристаллическим структурам – оригинальные журнальные статьи, опубликованные в открытой печати. Извлеченная кристаллоструктурная информация помещается (по специальному формату) в ASCII-файл с

последующей программной (на ПК) экспертизой по результатам расчета межатомных расстояний и других кристаллоструктурных характеристик и, в случае положительного решения, импортируется в ИВС специальными программами группового или одиночного импорта. Возможен импорт кристаллоструктурной информации через общепринятый в кристаллографии специализированный файл (Crystallographic Information File, CIF), содержащий кристаллоструктурные характеристики расшифрованной кристаллической фазы (формат – http://www.sdsc.edu/pb/cif/cif.html).

Базовая запись для индивидуального кристаллического вещества содержит информацию о названии (в соответствии с классификацией International Mineralogy Association или рекомендациями по наименованию неорганических веществ IUPAC), химическом составе, симметрии, параметрах элементарной ячейки, координатах атомных позиций с изотропными температурными факторами и заселенностями, информацию о межплоскостных расстояниях, HKL-индексах и интенсивностях сильнейших рефлексов рентгенодифракционной картины поликристалл-фазы, а также ссылки на соответствующие публикации по расшифровке или уточнению кристаллической структуры. Запись может быть специфицирована по полезным свойствам, особенностям химического состава и структуры, а также по PT-условиям синтеза. Каждая запись содержит «монокристальные» и «поликристальные» характеристики кристаллической фазы. Минералы классифицированы в соответствии с таксонами структурно-химической систематики минералов А.А. Годовикова, кристаллохимической классификации М. Чириотти, в настоящее время вводится классификация по структурным типам минералов (Бокий Г.Б.). Для 2000 фаз сделаны экспресс-оценки потенциальной энергии кристаллической решетки.

На базе вводимой информации локальный программный пакет эксперта позволяет автоматически сформировать вторую, производную от первой, базу расчетных поликристалл-стандартов, проводя синтез двух типов информации о кристаллической фазе. Связка «Кристаллическая структура фазы и ее расчетная поликристалл-рентгенограмма» является не только информационной основой WWW-MINCRYST, но и служит важнейшим инструментарием в руках пользователя (в особенности рентгеновского кристаллографа). Стержневая идея WWW-MINCRYST о синтезе двух типов кристаллоструктурной информации о кристаллической фазе реализует принципиально новый подход к формированию всей кристаллоструктурной информации о веществе и организации доступа к ней.

Для WWW-MINCRYST реализован импорт одобренной экспертами информации через web-интерфейс как в виде единичных записей, так и пакетов записей (до нескольких сот) с входным контролем, что позволяет проводить постоянную актуализацию информационного фонда. Также реализована возможность on-line редакции записей, их удаления, замены служебных файлов записей, архивации и восстановления базы данных через web-интерфейс.

2. Основные возможности ИВС WWW-MINCRYST

В ИВС реализована эффективная система поиска фаз по комплексу параметров, включая название минерала (полное или частичное), химический, или элементный, состав в различных комбинациях (присутствие или отсутствие элементов и их комбинации), симметрию, кристаллоструктурные характеристики, а также межплоскостные расстояния d(hkl), что в сочетании с химическим (элементным) составом дает возможность прямого

интерактивного качественного рентгенофазового анализа. В систему поиска добавлен поиск по классификационным параметрам нескольких кристаллохимических и структурных классификаций (Годовиков, Чириотти, Бокий). Система поиска по этим параметрам обеспечивает поиск (и группировку) минералов по указанным параметрам – низшим таксонам. По ряду параметров поиска WWW-MINCRYST до сих пор не имеет аналогов среди минералого-кристаллографических баз данных.

С помощью интегрированных в WWW-MINCRYST расчетных и презентационных модулей (в виде Java апплетов) последний превратился из первичной информационно-справочной в информационно-вычислительную систему, существенно расширив возможности пользователя по получению и представлению кристаллоструктурной и кристаллохимической информации о минеральном веществе. Разработанный и интегрированный (частично на конец 2012 года) в WWW-MINCRYST расчетный модуль WWW-Xraypol в связках с модулями WWW-Crystpic и WWW-Mixipol позволяет квалифицированному пользователю модифицировать структуры из информационного фонда (не затрагивая его), а затем рассчитывать и получать графические изображения моделей кристаллических структур и квазиреальных полных профилей расчетных поликристалл-рентгенограмм на основе временно извлекаемых и модифицируемых базовых BDM-файлов и тем самым решая задачи по модификации реальных структур.

Наличие большого количества структур и хороших средств визуализации позволяет с помощью WWW-MINCRYST развивать принципиально новые подходы к представлению кристаллических структур – благодаря гибкому использованию полиэдров, позволяющему формировать различные варианты структурных моделей минералов. Используется принцип: в ряде случаев кристаллическое пространство можно организовать в смешанном шаровом и полиэдрическом изображении на основе любых атомов в структуре, не строго привязываясь к традиционному катионно-анионному изображению.

Для всех записей через встроенный апплет WWW-Crystpic (Java-3D) доступны динамически создаваемые интерактивные изображения моделей кристаллических структур в шарах-сферах и в полиэдрических проекциях (до 138 позиций и до 1500(!) атомов на структуру) в соответствии с основными канонами минералогии и кристаллографии. Программа позволяет делать всевозможные манипуляции с моделью структуры, включая масштабирование, непрерывное и/или автоматическое дискретное вращение вокруг «экранных» осей X,Y,Z, ориентацию по кристаллографическим осям, hkl-фрагментацию структуры (на hkl-ориентированные фрагменты толщиной d(hkl)), наращивание элементарных ячеек вдоль любых выбранных направлений для формирования «сверхструктур» и мотивов, а также прямой «ручной» и автоматизированный для малых полиэдров (тетраэдров и октаэдров) расчет любых межатомных расстояний и углов в структуре. Программа изображает любые полиэдры, включая «дефектные» с необычно малыми («плохими») межатомными расстояниями. Более детальное описание всех возможностей апплета приведено здесь: http://mincryst.iem.ac.ru/rus/crystpic.php). В настоящее время из-за проблем с работой с библиотеками OpenGL (на которых реализован апплет) в составе новых версий Java рассматривается возможность переноса апплета на новые программные платформы типа встроенных средств HTML5

Также в WWW-MINCRYST интегрирован апплет WWW-Mixipol, предназначенный для графического представления полных расчетных спектральных профилей поликристалл-рентгенограмм с возможностями манипулирования спектрами для разных источников излучения и разных типов спектральных шкал. Также модуль способен формировать

рентгенограммы смесей фаз (до 6 фаз одновременно) при возможности варьирования относительными содержаниями компонентов смеси.

Как для структур, так и для спектров минералов предусмотрены упрощенные варианты представления (для старых браузеров и маломощных персоналок) в виде традиционных шаровых структур и линейчатых спектров.

В состав WWW-MINCRYST встроен модуль Xraypol, который в связках с программами WWW-Crystpic и WWW-Mixipol, позволяет квалифицированному пользователю менять «на лету» ряд параметров структуры и рассчитывать новые варианты изображений моделей кристаллических структур и квазиреальных полных профилей расчетных поликристалл-рентгенограмм на основе временно извлекаемых и модифицируемых базовых BDM-файлов, тем самым решая конкретные задачи по модификации реальных структур, но не затрагивая основной фонд WWW-MINCRYST.

Одними из первых среди научных интернет-ориентированных баз данных была разработана система динамически формируемых перекрестных веб-ссылок для связи записей с записями для конкретных минералов в ведущих минералогических базах данных, размещенных в Интернете. Система генерации динамических гиперссылок на внешние информационные ресурсы (в основном, на минералогические базы данных и поисковые системы) позволяет «прозрачно» для пользователя подключать большие внешние массивы данных, используя метод «генеральных» запросов. При этом пользователь сразу получает доступ к информации по интересующему его объекту, минуя стадии поиска или просмотра всей внешней базы. Кроме того, данный механизм реализует обратную связь, позволяя подобным же образом ссылаться этим базам уже на наши информационные объекты, что резко повышает востребованность WWW-MINCRYST внешними пользователями.

Наличие в WWW-MINCRYST более 8 тысяч кристаллических структур и встроенный универсальный расчетный комплекс позволили помимо ориентированных на поиск и предоставление информации возможностей использовать ИВС в разработке нетрадиционных научных подходов к интерпретации и представлению некоторых кристаллических структур. WWW Xraypol позволил, например, выявить в традиционных структурах возможность гибкого использования полиэдров и почти автоматически формировать различные варианты структурных моделей минералов. Как выяснилось, для части минералов кристаллическое пространство можно не строго привязываться к традиционному катионно-анионному изображению, а формировать структуры на основе любых атомов, входящих в ее состав. Метод особенно эффективен для сложных «неправильных» бескислородных структур (например, фосфиды, сложные сульфиды и др.).

Востребованность WWW-MINCRYST хорошо подтверждается статистикой обращений (7.5 млн. успешных единичных запросов за 2012 год, более 60 Гб скачанной информации, около 33000 уникальных сайтов), а также большим количеством отзывов, описаний и внешних ссылок на WWW-MINCRYST (см. раздел «Ссылки» на сайте).

О перспективах развития WWW-MINCRYST следует сказать отдельно. В связи с очень быстрым развитием web-технологий и (соответственно) браузеров, возникла необходимость кардинальной переработки нынешнего клиентского интерфейса, уже не отвечающего как новым технологиям (и зачастую несовместимого с новым ПО), так и требованиям пользователей. Она будет включать изменение способа представления записи путем сведения разобщенных сейчас полей (основные данные, CPDS карта, атомные позиции, структура, спектры, ссылки и др.) в единое информационное пространство, формируемое на базе динамического HTML (с использованием технологий Jason, DHTML, HTML-5, оверлейных структур). Предусматривается как переделка и рекомпиляция Java

апплетов (в связи с устареванием и неполной совместимостью кода с последними реализациями Java VM), так и их замена на модули, использующие мультимедийные технологии стандартов HTML-5 или Adobe Flash (с использованием оригинального авторского расчетного кода). Также будут изменены в сторону повышения дружелюбности и простоты) дизайн и элементы управления предоставляемой пользователю информации. В ходе проекта предусмотрено пополнение базы данных дополнительной сугубо минералогической информацией (фото минералов и рисунки кристаллографических форм и др.). Несмотря на то, что по ряду параметров поиска WWW-MINCRYST до сих пор не имеет аналогов, будет расширен круг потенциальных поисковых запросов (прежде всего в области поисков по составу и кристаллографическим данным).

Заключение

Разработанная и развиваемая на протяжении 15 лет ИВС WWW-MINCRYST является общедоступным, дружелюбным пользователю интерфейсом к большому объему кристаллоструктурной и кристаллохимической информации с развитыми средствами поиска, представления и обработки и может служить мощным инструментарием для всех исследователей в минералогии, кристаллографии, физике твердого тела, материаловедении и прочих смежных областях науки.

Проведение работ по ИВС WWW-MINCRYST в течение 1997–2012 годов было поддержано несколькими грантами РФФИ (в том числе в настоящее время – грантом РФФИ 12-07-00742-а, рук. Варламов Д.А.).

Литература

1. МИНКРИСТ – кристаллографическая база данных для минералов: локальный и сетевой (WWW) варианты / А.В. Чичагов, Д.А. Варламов, Р.А. Диланян и др. // Кристаллография. – 2001. – Т. 46. – № 5. – С.950–954.
2. Кристаллографическая и кристаллохимическая база данных для минералов и их структурных аналогов (WWW-MINCRYST) / А.В. Чичагов, Д.А. Варламов, Е.В. Ершов и др. // Записки Российского минералогического общества. – 2007. – Т. 136. – № 3. – С.135–141.
3. Чичагов, А.В. WWW-MINCRYST-2007 – Интернет-ориентированная база данных по кристаллографии/кристаллохимии минералов и их аналогов / А.В. Чичагов, Д.А. Варламов // «Научный сервис в сети Интернет: технологии параллельного программирования. 15 лет РФФИ», Труды Всероссийской научной конференции. – М.: изд-во МГУ, 2007. – С.390–392.

Варламов Дмитрий Анатольевич, с.н.с., Институт экспериментальной минералогии РАН, dima@iem.ac.ru

Докина Татьяна Николаевна, инженер-исследователь, Институт экспериментальной минералогии РАН, tdokina@mail.ru

Дрожжина Наталья Алексеевна, инженер-исследователь, Институт экспериментальной минералогии РАН, nadron@mail.ru

Самохвалова Ольга Леонидовна, инженер-исследователь, Институт экспериментальной минералогии РАН, olsamoh@mail.ru

WWW-MINCRYST: THE INTERNET-ORIENTED INFORMATION AND CALCULATION SYSTEM ON THE CRYSTALLOGRAPHY AND THE CRYSTAL CHEMISTRY OF MINERALS

D.A. Varlamov, Institute of Experimental Mineralogy RAS (Chernogolovka, Russian Federation),
T.N. Dokina, Institute of Experimental Mineralogy RAS (Chernogolovka, Russian Federation),
N.A. Drozhzhina, Institute of Experimental Mineralogy RAS (Chernogolovka, Russian Federation),
O.L. Samokhvalova, Institute of Experimental Mineralogy RAS (Chernogolovka, Russian Federation)

The Internet-oriented WWW-MINCRYST information-calculation system (ICS) intended for work with crystal structures of minerals, their synthetic analogs and elements is described. The ICS main components are actually a database (more than 8000 records for 3000 unique phases), supplied with a complex of means of search and information choice, means of multimedia submission of information (polyhedral and spherical interactive crystal structures, spectrum, etc.), opportunities of processing of spectral information, including using of the loaded user data. ICS is placed to the address http://mincryst.iem.ac.ru and is available to users without restrictions.

Keywords: Internet-oriented databases, crystal chemistry, crystal structures, minerals, PHP, MySQL

References

1. Chichagov A.V., Varlamov D.A., Dilanyan R.A., Dokina T.N., Drozhzhina N.A., Samokhvalova O.L., and Ushakovskaya T.V. MINCRYST: a Crystallographic Database for Minerals, Local and Network (WWW) Versions. Crystallography Reports, Vol. 46, Issue 5, P. 876–879.
2. Chichagov A.V., Varlamov D.A., Ershov Ye.V., Dokina T.N., Drozhzhina N.A., Samokhvalova O.L. Kristallograficheskaja i kristallohimicheskaja baza dannyh dlja mineralov i ih strukturnyh analogov (WWW-MINCRYST) [Crystallographic and crystal-chemical database for minerals and their structural analogues (WWW-MINCRYST)]. Zapiski Rossijskogo mineralogicheskogo obshhestva [Proceedings of the Russian Mineralogical Society]. 2007. Vol. 136. № 3. P. 135–141.
3. Chichagov A.V., Varlamov D.A. WWW-MINCRYST-2007 – Internet-orientirovannaya baza dannyh po kristallografii I kristallokhimii mineralov i ih analogov [WWW-MINCRYST-2007 – The Internet-oriented database on a crystallography and crystal chemistry of minerals and their analogs]. Trudy vserossiiskoi konferentsii «Nauchnyi servis v seti Internet: tekhnologii parallelnogo programmirovaniya, 15 let RFFI» (Novorossiisk, sentyabr 2007) [«Scientific service in Internet: technologies of parallel programming»: Proceedings of the Russian Scientific Conference (Novorossiysk, September 2007)]. Moscow. Moscow University Press, 2007. P. 390–392.

Поступила в редакцию 15 января 2013 г.

УДК 004.65

ОБ ОЦЕНКЕ КОММУНИКАЦИОННЫХ ЗАТРАТ ПРИ ОБРАБОТКЕ ФРАГМЕНТИРОВАННОГО ОТНОШЕНИЯ ДЛЯ РАВНОМЕРНОГО РАСПРЕДЕЛЕНИЯ

М.В. Губин, Л.Б. Соколинский

При обработке запросов в параллельных системах баз данных без совместного использования ресурсов в общем случае не удается избежать пересылок кортежей между процессорными узлами. В статье доказывается теорема, позволяющая получить оценку количества пересылаемых кортежей при обработке фрагментированного отношения для случая, когда функция пересылки функционально зависит от атрибута, значения которого распределены равномерно относительно атрибута фрагментации.

Ключевые слова: параллельные системы баз данных, архитектура без совместного использования ресурсов, фрагментный параллелизм, коммуникационные затраты.

Фрагментный параллелизм (data parallelism) [1] продолжает оставаться основной формой параллелизма в системах баз данных для многопроцессорных платформ с распределенной памятью. Фрагментный параллелизм предполагает разделение каждого отношения на части, называемые фрагментами, каждая из которых располагается на отдельном узле параллельной системы баз данных с архитектурой без совместного использования ресурсов [2]. Для разделения отношения на фрагменты используется *функция фрагментации* [3], отображающая множество кортежей отношения на множество узлов. Известно, что при использовании фрагментного параллелизма при обработке запросов в общем случае не удается избежать пересылок. При обработке фрагментированного отношения для организации пересылок кортежей используется *функция пересылки* (*распределения*) [3], которая для каждого кортежа вычисляет номер узла, на котором он должен быть обработан. Если этот номер не совпадает с номером хранения кортежа, кортеж пересылается.

Пересылки являются одной из самых затратных операций в параллельных системах баз данных без совместного использования ресурсов. В соответствии с этим является важным вопрос оценки количества пересылаемых кортежей [4]. В данной работе доказывается теорема, позволяющая получить такую оценку для случая, когда функция пересылки функционально зависит от атрибута, значения которого распределены равномерно относительно атрибута фрагментации.

Рассмотрим фрагментированное отношение R. Пусть $|R| = m$ – количество кортежей в R. Пусть отношение R разбито на k фрагментов с помощью *функции фрагментации* $\varphi : \ R \ \rightarrow \ \{0, 1, 2, \ldots, k-1\}$. Тогда фрагмент отношения R, хранящийся на j–том узле может быть определен следующим образом:

$$R_j = \{ \ r \mid \ r \in R, \ \ \varphi\,(r) = j\}. \tag{1}$$

Лемма 1. Пусть на отношении R задана функция фрагментации $\varphi : R \rightarrow \{0, 1, 2, \ldots, k-1\}$. Допустим, что существует взаимно-однозначное отображение $\gamma \ : \ R \ \rightarrow \ \{0, 1, 2, \ldots, m-1\}$ такое, что

$$\gamma\,(r) \ \bmod k = \varphi\,(r)\,, \forall r \in R\ . \tag{2}$$

Тогда размеры фрагментов $R_0, \ldots, R_k$ будут между собой *примерно равны*, то есть:

$$\lim_{m\to\infty} \frac{|R_i|}{|R_j|} = 1, \forall i, j \in \{0, 1, 2, \ldots, k{-}1\}. \tag{3}$$

Доказательство. Определим

$$M_j = \{\ l \mid\ \gamma^{-1}(l) \in R_j, 0 \le l{<}m\},\ j = 0, 1, \ldots, k{-}1, \tag{4}$$

тогда

$$|M_j| = |R_j|\,. \tag{5}$$

Определим

$$\varphi^{'}(l) = l \bmod k, \tag{6}$$

$$M_j^{'} = \{\ l \mid\ \varphi^{'}(l) = j, 0 \le l{<}m\},\ j = 0, 1, \ldots, k{-}1. \tag{7}$$

Покажем, что

$$M_j = M_j^{'},\ \ \forall j = 0, 1, \ldots, k{-}1.$$

Пусть сначала

$$l \in M_j. \tag{8}$$

Тогда из (4) следует $\gamma^{-1}(l) \in R_j$. Отсюда, в силу (1) получаем

$$\varphi(\gamma^{-1}(l)) = j.$$

Используя (2), отсюда, в свою очередь, получаем

$$\gamma\left(\gamma^{-1}(l)\right) \bmod k = j,$$

что эквивалентно

$$l \bmod k = j.$$

Сопоставляя это с (6), находим

$$\varphi^{'}(l) = j.$$

Учитывая (7), отсюда получаем

$$l \in M_j^{'},$$

откуда в силу (8) следует

$$M_j \subseteq M_j^{'}. \tag{9}$$

Пусть теперь

$$l \in M_j^{'}. \tag{10}$$

Учитывая (7), отсюда получаем

$$\varphi^{'}(l) = j.$$

Сопоставляя это с (6), находим

$$l \bmod k = j,$$

что эквивалентно

$$\gamma\left(\gamma^{-1}(l)\right) \mod k = j.$$

Используя (2), отсюда, в свою очередь, получаем

$$\varphi\left(\gamma^{-1}(l)\right) = j.$$

В силу (1), из последнего равенства следует

$$\gamma^{-1}(l) \in R_j.$$

Тогда из (4) следует

$$l \in M_j.$$

Учитывая (10), отсюда получаем

$$M_j^{'} \subseteq M_j,$$

откуда в силу (9) следует

$$M_j^{'} = M_j.$$

В сочетании с (5) это дает

$$\left|M_j^{'}\right| = \left|R_j\right|, \ \forall j \in \{0,1,2,\ldots,k{-}1\} \tag{11}$$

Таким образом, утверждение (3) равносильно утверждению

$$\lim_{m\to\infty} \frac{\left|M_i^{'}\right|}{\left|M_j^{'}\right|} = 1, \ \forall i,j \in \{0,1,2,\ldots,k{-}1\}. \tag{12}$$

Докажем это утверждение. В силу (6) и (7) имеем

$$M_j^{'} = \{\ l \ \mid \ l \bmod k = j, 0 \le l{<}m\}, \ \ j = 0,1,\ldots,k{-}1.$$

Отсюда непосредственно следует[1]

$$\left\lfloor \frac{m}{k} \right\rfloor \le \left|M_j^{'}\right| \le \left\lceil \frac{m}{k} \right\rceil, \ \forall j \in \{0,1,2,\ldots,k{-}1\}.$$

Тогда

$$\frac{\left\lfloor \frac{m}{k} \right\rfloor}{\left\lceil \frac{m}{k} \right\rceil} \le \frac{\left|M_i^{'}\right|}{\left|M_j^{'}\right|} \le \frac{\left\lceil \frac{m}{k} \right\rceil}{\left\lfloor \frac{m}{k} \right\rfloor}, \ \forall i,j \in \{0,1,2,\ldots,k{-}1\}. \tag{13}$$

Имеем

$$\lim_{m\to\infty} \frac{\left\lfloor \frac{m}{k} \right\rfloor}{\left\lceil \frac{m}{k} \right\rceil} = \lim_{m\to\infty} \frac{\left\lceil \frac{m}{k} \right\rceil}{\left\lfloor \frac{m}{k} \right\rfloor} = 1.$$

Вместе с (13) это доказывает справедливость утверждения (12), а вместе с ним и утверждения (3). □

[1]Здесь без ограничения общности мы можем полагать, что m > k.

Определение 1. Пусть задана функция фрагментации

$$\varphi : R(A_1, \ldots, A_n) \to \{0, 1, 2, \ldots, k-1\}.$$

Будем говорить, что φ *функционально зависит* от атрибута $A_i (0 < i \leq n)$, если для любых $r,\ \acute{r} \in R$ таких, что $\pi_{A_i}(r) = \pi_{A_i}(\acute{r})$, имеем $\varphi(r) = \varphi(\acute{r})$.

Определение 2. Пусть имеется отношение $R(\ldots, A, \ldots)$. Пусть D_A – множество всех значений атрибута A, присутствующих в R. Обозначим через $T(R, A, a)$ количество кортежей в отношении R, у которых атрибут A принимает значение a. Будем говорить, что значения атрибута A *равномерно распределены* в R, если для любых $a, \tilde{a} \in D_A$ имеем

$$T(R, A, a) = T(R, A, \tilde{a}). \tag{14}$$

Лемма 2. Пусть значения атрибута A равномерно распределены в $R(\ldots, A, \ldots)$, $m = |R|$ кратно $V(R, A)$[2], а $V(R, A)$ кратно $k > 0$. Тогда существует функционально зависящая от A функция фрагментации φ, которая обеспечивает равномерное распределение кортежей по фрагментам:

$$\lim_{m \to \infty} \frac{|R_i|}{|R_j|} = 1,\ \forall i, j \in \{0, 1, 2, \ldots, k-1\}. \tag{15}$$

Доказательство. Выполним сортировку кортежей R в порядке возрастания значений атрибута A:

$$R = \{r_0, r_1, \ldots, r_{m-1}\}, \tag{16}$$

где $\pi_A(r_i) \leq \pi_A(r_{i+1})$ для всех $i = 0, \ldots, m-2$.

Здесь $\pi_A(r)$ обозначает значение атрибута A в кортеже r. Обозначим через $\eta(r)$ порядковый номер кортежа r в последовательности (16). По определению 2, для любых $d, \tilde{d} \in D_A$ имеем:

$$T(R, A, d) = T(R, A, \tilde{d}). \tag{17}$$

Положим,

$$c = \frac{m}{k}. \tag{18}$$

Так как по условию леммы m кратно k, то $c \in \mathbb{N}$. Определим на множестве целых неотрицательных чисел функцию

$$\overline{\gamma}(i) = \left\lfloor \frac{i}{c} \right\rfloor + (i \bmod c)\, k, \quad i \in \mathbb{Z}_{\geq 0}. \tag{19}$$

Положим,

$$\gamma(r) = \overline{\gamma}(\eta(r)), \quad \forall r \in R. \tag{20}$$

Покажем, что γ является взаимно-однозначным отображением множества кортежей отношения R в множество $\{0, \ldots, m-1\}$. Так как η является взаимно-однозначным отображением множества кортежей отношения R в множество $\{0, \ldots, m-1\}$, нам достаточно показать, что

[2]Под $V(R, A)$ понимается количество различных значений в пределах столбца-атрибута A отношения R.

ограничение $\overline{\gamma}$ на множество $\{0,\ldots,m{-}1\}$ является взаимно-однозначным отображением множества $\{0,\ldots,m{-}1\}$ в себя. Сначала покажем, что

$$0 \le \overline{\gamma}(i) \le m-1, \quad \forall i \in \{0,\ldots,m{-}1\}. \tag{21}$$

Из (19) непосредственно следует

$$\min_{0\le i<m} \overline{\gamma}(i) = 0.$$

Далее имеем

$$\begin{aligned}\max_{0\le i<m}(\overline{\gamma}(i)) &= \max_{r\in R}\lfloor i/c\rfloor + \max_{r\in R}((\eta(r) \bmod c)\,k)\\ &= \left\lfloor \frac{1}{c}\max_{r\in R}\eta(r)\right\rfloor + k\max_{r\in R}(\eta(r) \bmod c)\\ = \left\lfloor \frac{1}{c}(m-1)\right\rfloor + k(c-1) &= \left\lfloor \frac{k}{m}(m-1)\right\rfloor + k\left(\frac{m}{k}-1\right)\\ &= \left\lfloor k-\frac{1}{m}\right\rfloor + m - k = k-1+m-k = m-1.\end{aligned}$$

Таким образом, (21) имеет место.

Теперь для доказательства биективности отображения $\overline{\gamma}$ достаточно доказать, что существует обратная функция $\overline{\gamma}^{-1}$, определенная на множестве неотрицательных чисел такая, что

$$\overline{\gamma}^{-1}(\overline{\gamma}(i)) = i, \quad \forall i \in \{0,\ldots,m{-}1\}. \tag{22}$$

Положим, $\overline{\gamma}^{-1}(j) = \left\lfloor \frac{j}{k}\right\rfloor + (j \ mod\ k)\,c$. Тогда

$$\overline{\gamma}^{-1}(\overline{\gamma}(i)) = \left\lfloor \frac{\overline{\gamma}(i)}{k}\right\rfloor + (\overline{\gamma}(i) \bmod k)\,c.$$

В силу (19) отсюда получаем

$$\overline{\gamma}^{-1}(\overline{\gamma}(i)) = \left\lfloor \frac{\lfloor \frac{i}{c}\rfloor + (i \bmod c)\,k}{k}\right\rfloor + \left((\left\lfloor \frac{i}{c}\right\rfloor + (i \bmod c)\,k) \bmod k\right)c.$$

Перегруппировав, получим

$$\overline{\gamma}^{-1}(\overline{\gamma}(i)) = \left\lfloor \frac{\lfloor \frac{i}{c}\rfloor}{k} + (i \bmod c)\right\rfloor + \left(\left\lfloor \frac{i}{c}\right\rfloor \bmod k\right)c. \tag{23}$$

Из (18) следует

$$\left\lfloor \frac{i}{c}\right\rfloor = \left\lfloor \frac{ik}{m}\right\rfloor.$$

Поскольку $0 \le i < m$, отсюда получаем

$$\left\lfloor \frac{i}{c}\right\rfloor < k,$$

откуда, в свою очередь, следует

$$0 \le \frac{\lfloor \frac{i}{c}\rfloor}{k} < 1. \tag{24}$$

В силу этого (23) равносильно

$$\overline{\gamma}^{-1}\left(\overline{\gamma}\left(i\right)\right) = \left(i \bmod c\right) + \left(\left\lfloor \frac{i}{c} \right\rfloor \bmod k\right) c. \tag{25}$$

По определению операции mod

$$x \bmod y = x - \left\lfloor \frac{x}{y} \right\rfloor y.$$

Используя эту формулу, мы можем преобразовать (25) к виду

$$\overline{\gamma}^{-1}\left(\overline{\gamma}\left(i\right)\right) = i - \left\lfloor \frac{i}{c} \right\rfloor c + \left(\left\lfloor \frac{i}{c} \right\rfloor - \left\lfloor \frac{\left\lfloor \frac{i}{c} \right\rfloor}{k} \right\rfloor k\right) c. \tag{26}$$

В силу (24) имеем

$$\left\lfloor \frac{\left\lfloor \frac{i}{c} \right\rfloor}{k} \right\rfloor = 0.$$

Учитывая этот факт, (26) эквивалентно

$$\overline{\gamma}^{-1}\left(\overline{\gamma}\left(i\right)\right) = i - \left\lfloor \frac{i}{c} \right\rfloor c + \left\lfloor \frac{i}{c} \right\rfloor c,$$

то есть

$$\overline{\gamma}^{-1}\left(\overline{\gamma}\left(i\right)\right) = i.$$

Следовательно, обратная функция для $\overline{\gamma}$ существует, то есть $\overline{\gamma}$ биективно. С учетом (21) отсюда получаем, что ограничение $\overline{\gamma}$ на множество $\{0,\ldots,m-1\}$ является взаимно-однозначным отображением множества $\{0,\ldots,m-1\}$ в себя. А это означает, что γ является взаимно-однозначным отображением множества кортежей отношения R в множество $\{0,\ldots,m-1\}$.

Определим

$$\varphi\left(r\right) = \gamma\left(r\right) \bmod k. \tag{27}$$

Покажем, что φ функционально зависит от A, то есть, если $\pi_A\left(r\right) = \pi_A\left(\acute{r}\right)$, то $\varphi\ \left(r\right) = \varphi\ \left(\acute{r}\right)\ \ \forall r, \acute{r} \in R$. Пусть

$$\pi_A\left(r\right) = \pi_A\left(\acute{r}\right). \tag{28}$$

По условию леммы $V(R,A)$ кратно k, то есть существует $v \in \mathbb{N}$ такой, что

$$v = V\left(R,A\right)/k. \tag{29}$$

Сначала разобьем последовательность (16) на подпоследовательности

$$G_0 = \left\{r_0,\ldots,r_{\left(\frac{m}{V(R,A)}\right)-1}\right\},\ \ldots, G_{V(R,A)-1} = \left\{r_{\frac{(V(R,A)-1)m}{V(R,A)}},\ldots,r_{m-1}\right\} \tag{30}$$

обладающие свойствами:

$$\begin{aligned} &1)\ \pi_A\left(r\right) = \pi_A\left(\acute{r}\right)\ \ \forall r, \acute{r} \in G_l;\\ &2)\ \pi_A\left(r\right) \neq \pi_A\left(\acute{r}\right)\ \ \forall r \in G_l, \acute{r} \notin G_l;\\ &3)\ |G_l| = \tfrac{m}{V(R,A)}; \end{aligned} \tag{31}$$

для любого и целого $l \in [0, V(R,A)-1]$.

Затем разобьем последовательность (16) на следующие подпоследовательности:

$$Q_0 = \left\{ r_0, \ldots, r_{\left(\frac{v \cdot m}{V(R,A)}\right)-1} \right\}, \quad \ldots, Q_{k-1} = \left\{ r_{\frac{(V(R,A)-1)\cdot v \cdot m}{V(R,A)}}, \ldots, r_{m-1} \right\} \tag{32}$$

обладающие свойствами:

$$\begin{array}{l} 1)\ \forall l \in [0, V(R,A)-1]\ \ \exists\ u \in [0, k-1] : G_l \subset Q_u; \\ 2)\ \ |Q_u| = \frac{m}{k}. \end{array} \tag{33}$$

Отметим, что свойство (33) обеспечивается условием (29). Подставляя вместо v правую часть равенства (29) преобразуем (32) к виду:

$$Q_0 = \left\{ r_0, \ldots, r_{\left(\frac{m}{k}\right)-1} \right\}, \quad \ldots, Q_{k-1} = \left\{ r_{\frac{(V(R,A)-1)\cdot m}{k}}, \ldots, r_{m-1} \right\}. \tag{34}$$

Более кратко это можно записать следующим образом:

$$Q_u = \left\{ r_{\frac{m}{k}u}, \ldots, r_{\frac{m}{k}(u+1)-1} \right\},\ u \in [0, k-1]. \tag{35}$$

Выберем $r, \acute{r} \in R$ такие, что $\pi_A(r) = \pi_A(\acute{r})$. Тогда, в силу (31) имеем $r, \acute{r} \in G_l$ для некоторого $l \in [0, V(R,A)-1]$. По свойству (33), отсюда следует, что $r, \acute{r} \in Q_u$ для некоторого $u \in [0, k-1]$. Положим

$$i = \eta(r), \quad j = \eta(\acute{r}), \tag{36}$$

тогда в силу (35) имеем

$$i, j \in \left[u\frac{m}{k}; \frac{m}{k}(u+1) - 1 \right], \quad 0 \le u < k. \tag{37}$$

Отсюда следует

$$\frac{i}{m/k}, \frac{j}{m/k} \in \left[u; u + 1 - \frac{1}{m/k} \right], \quad 0 \le u < k.$$

Так как u является целым неотрицательным числом, из предыдущего получаем

$$\left\lfloor \frac{i}{m/k} \right\rfloor, \left\lfloor \frac{j}{m/k} \right\rfloor \in \left[u; u + 1 - \frac{1}{m/k} \right], \quad 0 \le u < k.$$

Поскольку $0 < \frac{1}{m/k} < 1$, то отсюда, в свою очередь, получаем

$$\left\lfloor \frac{i}{m/k} \right\rfloor, \left\lfloor \frac{j}{m/k} \right\rfloor \in [u; u], \quad 0 \le u < k,$$

а это означает, что

$$\left\lfloor \frac{i}{m/k} \right\rfloor = \left\lfloor \frac{j}{m/k} \right\rfloor = u, \quad 0 \le u < k. \tag{38}$$

В силу (27) имеем

$$\varphi(r) = \gamma(r) \mod k.$$

Учитывая (20), это эквивалентно

$$\varphi(r) = \overline{\gamma}(\eta(r)) \mod k.$$

Используя (36), преобразуем последнее равенство к виду

$$\varphi(r) = \overline{\gamma}(i) \mod k.$$

Учитывая (19) и (18), отсюда получаем

$$\varphi(r) = \left(\left\lfloor \frac{i}{m/k} \right\rfloor + \left(i \bmod \frac{m}{k} \right) k \right) \mod k.$$

Второе слагаемое кратно k и по модулю k, поэтому оно будет равно нулю. Следовательно

$$\varphi(r) = \left\lfloor \frac{i}{m/k} \right\rfloor \mod k. \tag{39}$$

Аналогичным образом получаем

$$\varphi(\acute{r}) = \left\lfloor \frac{j}{m/k} \right\rfloor \mod k. \tag{40}$$

С учетом (38), из (39) и (40) следует $\varphi(r) = \varphi(\acute{r})$, то есть φ функционально зависит от A.

Остается заметить, что функции φ и γ удовлетворяют условиям леммы 1, из которой следует, что

$$\lim_{m\to\infty} \frac{|R_i|}{|R_j|} = 1, \forall i, j \in \{0, 1, 2, \ldots, k-1\}.$$

□

Определение 3. Пусть имеется отношение $R(A, B, \ldots)$. Пусть $D_A = \{a_0, \ldots, a_{V(R,A)-1}\}$ – множество всех значений атрибута A, присутствующих в R. Обозначим $R_u^A = \sigma_{A=a_u}(R)$, $u \in \{0, \ldots, V(R,A)-1\}$. Пусть $D_B = \{b_0, \ldots, b_{V(R,B)-1}\}$ – множество всех значений атрибута B, присутствующих в R. Будем говорить, что значения атрибута B *распределены равномерно относительно атрибута* A, если для любого $b \in D_B$ имеем

$$T\left(R_0^A, B, b\right) = T\left(R_1^A, B, b\right) = \cdots = T\left(R_{V(R,A)-1}^A, B, b\right). \tag{41}$$

Теорема. Пусть имеется отношение $R(A, B, \ldots)$, в котором значения атрибута A распределены равномерно, а значения атрибута B, в свою очередь, распределены равномерно относительно атрибута A. Положим $m = |R|$. Пусть m кратно $V(R,A)$, а $V(R,A)$ кратно k, где k – количество фрагментов, на которое необходимо разбить R ($k > 0$). Тогда существует функция фрагментации φ, функционально зависящая от атрибута A, такая, что для любой функции пересылки ψ, функционально зависящей от атрибута B, количество пересылаемых кортежей равно $(1 - 1/k)m$.

Доказательство. Выберем в качестве функции фрагментации φ такую функцию, функционально зависящую от атрибута A, которая обеспечивает равномерное распределение кортежей по фрагментам. Такая функция существует в силу леммы 2. Обозначим через R_j j-тый фрагмент отношения R:

$$R_j = \{r \mid r \in R, \varphi(r) = j\}.$$

Пусть $D_A = \{a_0,\ \ldots, a_{V(R,A)-1}\}$ – множество всех значений атрибута A, присутствующих в R. Обозначим $R_u^A = \sigma_{A=a_u}(R)$, $u = 0,\ \ldots, V(R,A)-1$. Так как по условию теоремы значения атрибута A распределены равномерно в R, имеем

$$\left|R_u^A\right| = \frac{m}{V(R,A)} \qquad (u = 0,\ \ldots, V(R,A)-1)\,. \tag{42}$$

Поскольку функция фрагментации φ функционально зависит от A, то

$$\forall u \in \{0,\ \ldots, V\,(R,A)\,{-}1\} \quad \exists j \in \{0,\ \ldots, k{-}1\} \ : \quad R_u^A \subset R_j.$$

В силу того, что φ обеспечивает равномерное распределение кортежей по фрагментам, отсюда следует, что мы можем перенумеровать элементы множества $\left\{R_u^A \;\middle|\; u = 0,\ \ldots, V(R,A)-1\right\}$ таким образом, что

$$R_j = \bigcup_{u=\frac{V(R,A)}{k}j}^{\frac{V(R,A)}{k}(j+1)-1} R_u^A \qquad (j = 0,\ldots,k{-}1). \tag{43}$$

Причем

$$\forall\ u \neq v\ : R_u^A \cap R_v^A = \emptyset, \tag{44}$$

и

$$\forall\ j \in \{0,\ \ldots,k{-}1\}\ \forall\ u \in \{0,\ \ldots, \mathrm{V\,(R,A)}\,{-}1\} : \quad |R_j| = \frac{V\,(R,A)}{k}\left|R_u^A\right|. \tag{45}$$

Так как по условию теоремы значения атрибута B распределены равномерно относительно атрибута A, то из (41) непосредственно следует, что

$$\left|\sigma_{B=b}\left(R_0^A\right)\right| = \left|\sigma_{B=b}\left(R_1^A\right)\right| = \ \ldots = \left|\sigma_{B=b}\left(R_{V(R,A)-1}^A\right)\right| \quad (\forall b \in D_B)\,.$$

Отсюда и из (43)-(45) получаем

$$|\sigma_{B=b}\,(R_0)| = |\sigma_{B=b}\,(R_1)| = \ \ldots = |\sigma_{B=b}\,(R_{k-1})| \quad (\forall b \in D_B)\,. \tag{46}$$

Пусть $D_B = \{b_0,\ \ldots, b_{V(R,B)-1}\}$ – множество всех значений атрибута B, присутствующих в R. Так как по условию теоремы функция пересылки ψ функционально зависит от атрибута B, то существует функция $\overline{\psi} : D_B \to \{0,\ldots,k-1\}$ такая, что

$$\forall r \in R : \psi\,(r) = \overline{\psi}\,(r.B)\,. \tag{47}$$

Обозначим $R_v^B = \sigma_{B=b_v}\,(R)$, $v = 0,\ \ldots, V(R,B)-1$. Имеем

$$R = \bigcup_{0 \le v < V(R,B)} R_v^B, \tag{48}$$

причем

$$\forall v \neq \acute{v}\colon \quad R_v^B \cap R_{\acute{v}}^B = \emptyset. \tag{49}$$

Имеем

$$\forall r \in R_v^B\colon \psi\,(r) = \overline{\psi}\,(b_v)\,.$$

Это означает, что кортежи множества $\sigma_{B=b_v}\left(R_{\overline{\psi}(b_v)}\right)$ пересылаться не будут. Учитывая (46), заключаем, что из множества R_v^B пересылке не подвергнутся в точности $|\sigma_{B=b_v}(R_0)|$ кортежей. Используя (48) и (49), отсюда получаем, что количество кортежей из R, которые не подвергнутся пересылке, равно

$$\sum_{v=0}^{V(R,B)-1} |\sigma_{B=b_v}(R_0)| = |R_0|.$$

В силу того, что φ обеспечивает равномерное распределение кортежей по фрагментам, имеем $|R_0| = \frac{m}{k}$. Отсюда следует, что количество пересылаемых кортежей равно

$$m - \frac{m}{k} = \left(1 - \frac{1}{k}\right) m.$$

□

Работа выполнена при поддержке гранта РФФИ №12-07-00443-а.

Литература

1. Rahm, E. Parallel Query Processing in Shared Disk Database Systems / Rahm E. // ACM SIG-MOD Record. – 1993. – Vol. 22, No. 4. – P. 32–37.
2. Соколинский, Л.Б. Обзор архитектур параллельных систем баз данных / Соколинский Л.Б. // Программирование. – 2004. – No. 6. – С. 49–63.
3. Лепихов, А.В. Обработка запросов в СУБД для кластерных систем / Лепихов А.В., Соколинский Л.Б. // Программирование. – 2010. – No 4. – С. 25–39.
4. Hasan, W. Coloring Away Communication in Parallel Query Optimization / Hasan W., Motwani R. // VLDB'95, Proceedings of 21st International Conference on Very Large Data Bases, September 11–15, 1995, Zurich, Switzerland. – Morgan Kaufmann, 1995. – P. 239–250.

Максим Владимирович Губин, кафедра технологии машиностроения, станков и инструментов, Южно-Уральский государственный университет (г. Златоуст, Российская Федерация), paragun@yandex.ru.

Леонид Борисович Соколинский, доктор физико-математических наук, профессор, кафедра системного программирования, Южно-Уральский государственный университет (г. Челябинск, Российская Федерация), sokolinsky@acm.org.

ABOUT COMMUNICATION COST ESTIMATION FOR PROCESSING OF PARTITIONED RELATION WITH UNIFORM DISTRIBUTION

M.V. Gubin, South Ural State University (Zlatoust, Russian Federation),
L.B. Sokolinky, South Ural State University (Chelyabinsk, Russian Federation)

Query processing in the shared-nothing parallel database systems demands a data exchange between processor nodes. In the paper, we present a theorem, which gives estimation for the amount of tuples which have to be transferred during processing partitioned relation. We consider the case when the transfer function is functionally dependent on an attribute, which is uniformly distributed relative to the partitioning attribute.

Keywords: parallel database system, shared-nothing architecture, partitioning parallelism, communication overhead.

References

1. Rahm E. Parallel Query Processing in Shared Disk Database Systems. ACM SIG-MOD Record. 1993. Vol. 22, No. 4. P. 32–37.
2. Sokolinky L.B. Survey of Architectures of Parallel Database Systems. Programming and Computer Software. 2004. Vol. 30, No. 6. P. 337–346.
3. Lepikhov A.V., Sokolinky L.B. Query Processing in a DBMS for Cluster Systems. Programming and Computer Software. 2010. Vol. 36, No. 4. P. 205–215.
4. Hasan W., Motwani R. Coloring Away Communication in Parallel Query Optimization. VLDB'95, Proceedings of 21st International Conference on Very Large Data Bases, September 11–15, 1995, Zurich, Switzerland. Morgan Kaufmann. 1995. P. 239–250.

Поступила в редакцию 18 июля 2012 г.

УДК 004.891.3

СИСТЕМА ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ ФИЗИОЛОГИЧЕСКИХ ИССЛЕДОВАНИЙ В СПОРТЕ ВЫСШИХ ДОСТИЖЕНИЙ

В.В. Епишев, А.П. Исаев, Р.М. Миниахметов, А.В. Мовчан, А.С. Смирнов, Л.Б. Соколинский, М.Л. Цымблер, В.В. Эрлих

В работе представлена архитектура системы MedMining, которая предназначена для интеллектуального анализа данных физиологических исследований спортсменов. Система обеспечивает экспорт результатов измерений в хранилище данных. Поддерживается хранение как необработанных результатов измерений (значений, поступающих непосредственно с приборов), так и их обработанных аналогов (получаемых путем усреднения, аппроксимации или других интегрирующих действий над обработанными данными). Интеллектуальный анализ результатов измерений направлен на определение ключевых показателей результативности и эффективности методики тренировок, а также поиск трендов и аномалий в этих показателях для гибкого изменения тренировочного графика.

Ключевые слова: интеллектуальный анализ данных, хранилище данных, физиологические исследования, спорт высших достижений

Введение

Под *интеллектуальным анализом данных (Data Mining)* понимают совокупность методов для обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности [2]. Технологии интеллектуального анализа данных применяются в широком спектре предметных областей, в том числе в медицине и физиологии [1, 3]. Одной из сфер приложения технологий интеллектуального анализа данных в медицине и физиологии является спорт высших достижений. Мониторинг, накопление и интеллектуальный анализ данных о физической и биохимической активности спортсменов направлены на определение ключевых показателей результативности и эффективности методики тренировок и поиск трендов и аномалий в этих показателях для гибкого изменения тренировочного графика [4, 10–12].

В рамках программы развития Южно-Уральского национального исследовательского государственного университета (НИУ ЮУрГУ) кафедрой теории и методики физической культуры и спорта и кафедрой системного программирования выполняется совместный проект, в рамках которого разрабатывается программная система MedMining, предназначенная для сбора и интеллектуального анализа данных физиологических исследований спортсменов НИУ ЮУрГУ. В данной статье рассматривается контекст данного проекта, архитектура и принципы реализации системы MedMining.

Статья организована следующим образом. Раздел 1 содержит описание предметной области проекта MedMining. В разделе 2 рассмотрена архитектура системы и технологический цикл работы с ней. В заключении суммируются полученные результаты и обсуждаются направления дальнейших исследований.

1. Предметная область проекта MedMining

Модель предметной области описывает организацию и проведение исследований, связи между спортсменами и тренерами, а также используемое оборудование. Диаграмма «сущность-связь», отражающая модель предметной области, представлена на рис. 1.

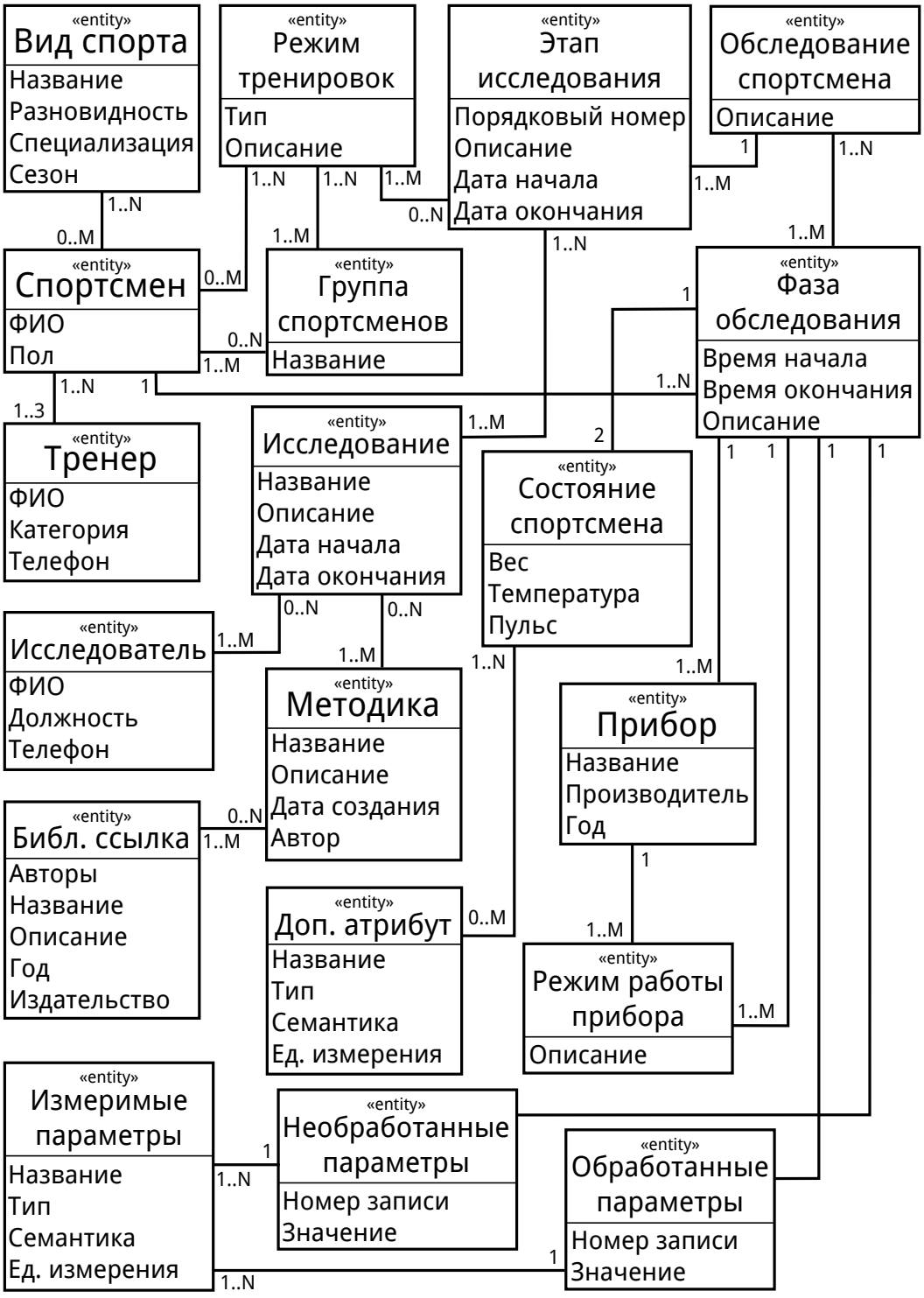

Рис. 1. Модель предметной области

1.1. Организация исследований

Организация исследований характеризуется следующими сущностями: исследование, исследователь, методика, библиографическая ссылка и этап исследования.

Исследование направлено на оценку состояния и подготовленности спортсменов на базе комплекса методик тренировки.

Исследователь — сотрудник, осуществляющий подготовку приборов, наблюдение за спортсменом во время исследования и экспорт данных исследования.

Методика представляет собой описание подхода к проведению исследований и подразумевает проведение набора исследований.

Библиографическая ссылка — библиографический источник, использованный в разработанной методике.

Этап исследования — одна из последовательных задач исследования, которая связана с оценкой физиологического состояния спортсмена.

Лаборант проводит измерение различных показателей различных показателей спортсменов физиологического состояния спортсменов (пульс, давление, ЭКГ и др.) во время и после тренировок с помощью различного оборудования.

1.2. Спортсмены и тренеры

Связи между спортсменами и тренерами характеризуются следующими сущностями: спортсмен, группа спортсменов, режим тренировок, вид спорта, тренер.

Спортсмен — испытуемый в исследованиях. Может заниматься несколькими видами спорта под руководством нескольких тренеров.

Группа спортсменов — спортсмены, которые имеют одинаковый режим тренировок.

Режим тренировок — описание тренировочного процесса спортсмена или группы спортсменов в рамках одного этапа исследования.

Вид спорта — название и характеристики определенного вида спорта.

Тренер — ответственный за подготовку спортсменов. Может тренировать различных спортсменов по разным видам спорта.

1.3. Проведение исследований

Проведение исследований характеризуется следующими сущностями: обследование спортсмена, фаза обследования, состояние спортсмена, дополнительный атрибут, необработанное данное и обработанное данное.

Обследование спортсмена — снятие физиологических показателей спортсмена на приборах в рамках одного этапа исследования.

Фаза обследования — шаг обследования, предполагающий снятие показаний определенного прибора в ходе тренировки спортсмена.

Состояние спортсмена — набор атрибутов, характеризующих динамичное физиологическое состояние спортсмена.

Дополнительный атрибут — атрибут сущности «Состояние спортсмена», название и семантика которого определяется исследователем. Может быть введено неограниченное количество дополнительных атрибутов. Дополнительный атрибут позволяет исследователю

отслеживать влияние различных факторов любой природы на физиологическое состояние спортсмена тренировочный процесс (например, факт приема и количество принятых препаратов до или после тренировки).

Необработанное данное — единичное значение одного из параметров, измеряемых прибором.

Обработанное данное — значение, получаемое из необработанных данных путем усреднения, аппроксимации или других интегрирующих действий.

1.4. Оборудование

Оборудование, используемое при проведении исследований, описывается сущностями прибор, режим работы прибора, измеряемый параметр.

Прибор представляет собой программно-аппаратный комплекс для измерения физиологических параметров спортсмена.

Режим работы прибора — состояние работы прибора, при котором он может измерять определенную группу параметров.

Измеряемый параметр — параметр физиологического состояния спортсмена, который можно измерить данным прибором.

2. Архитектура системы MedMining

Описание архитектуры системы приводится в нотации UML, и включает в себя описание аппаратных компонент и программной архитектуры.

2.1. Аппаратные компоненты системы

На рис. 2 представлена диаграмма развертывания системы MedMining.

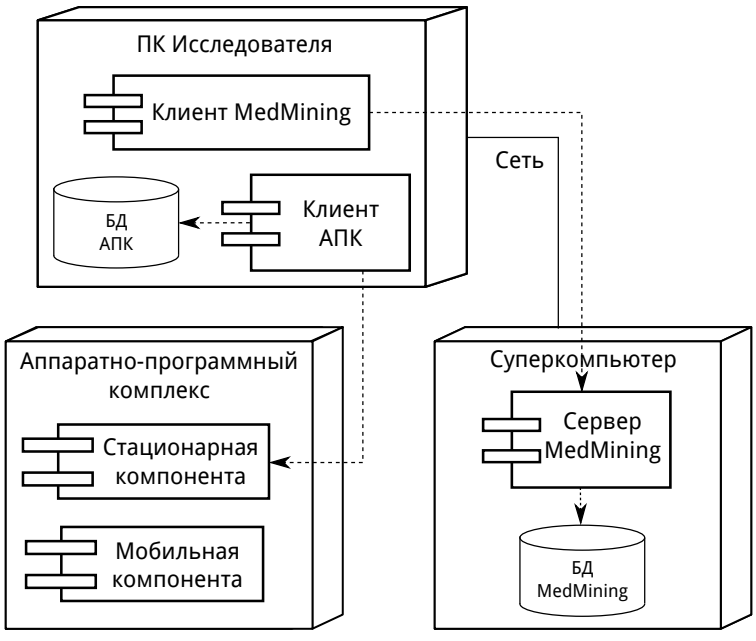

Рис. 2. Размещение системы MedMining

Система MedMining размещается на трех основных компонентах: с помощью *аппаратно-программного комплекса (АПК)* Исследователь проводит снятие физиологических показателей спортсменов, для взаимодействия с АПК и сервером системы MedMining, исследователь использует обычный *персональный компьютер (ПК)*, для проведения ресурсоемких вычислений, таких как интеллектуальный анализ данных сверхбольших объемов, используется *суперкомпьютер* «СКИФ-Аврора ЮУрГУ» [5].

На диаграмме деятельности (рис. 3) показано взаимодействие актеров с основными компонентами системы.

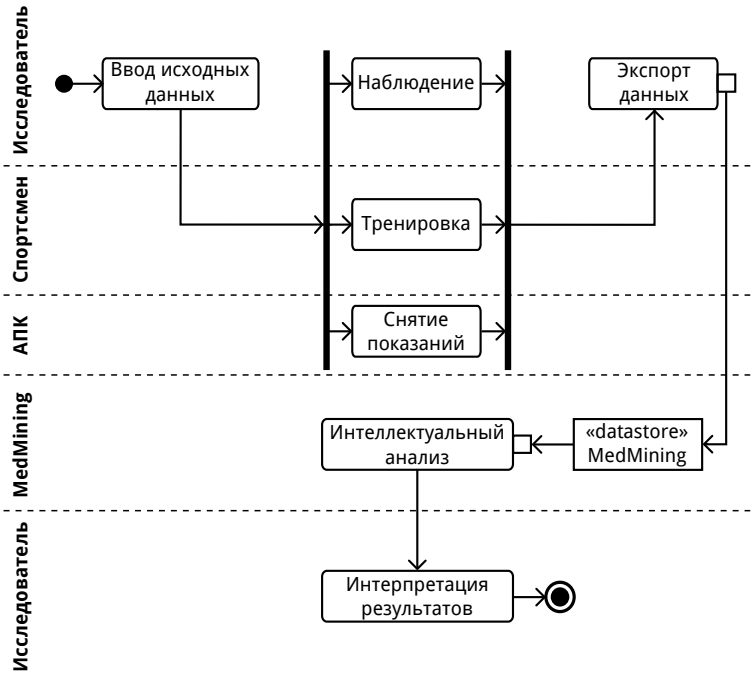

Рис. 3. Технологический цикл проведения исследований

В процессе снятия показаний задействованы следующие актеры: Исследователь, Спортсмен, АПК и система MedMining. Перед началом каждой Фазы обследования (тренировки) Исследователь выполняет снятие показаний Состояния спорстмена и Дополнительных атрибутов, которые относятся к данному Исследованию. Полученные показатели Исследователь вносит в систему MedMining.

После калибровки оборудования АПК спортсмен начинает тренировку. Во время тренировки Исследователь следит за состоянием Спорстмена. Показатели физиологического состояния спорстмена поступают в ПО АПК, на котором проходит данная фаза обследования. По окончании тренировки Исследователь выполняет выгрузку обработанных и необработанных данных исследования и вносит их в систему MedMining. Полученные данные хранятся на сервере системы MedMining — как в виде текстовых файлов, так и в виде таблиц хранилища.

Собранные данные подвергаются интеллектуальному анализу по выбору Исследователя. Система MedMining обеспечивает решение следующих основных задач интеллектуального анализа данных: кластеризация, классификация и анализ ассоциативных правил.

2.2. Программные компоненты системы

Модульная структура системы MedMining представлена на рис. 4 и содержит следующие компоненты. *Внешний интерфейс (FrontEnd)* включает в себя подсистемы, обеспечивающие интерфейс конечного пользователя и передачу полученной от него информации подсистемам *внутреннего интерфейса (BackEnd)*, которые выполняют ее обработку.

В состав BackEnd входят следующие подсистемы. Свободная реляционная СУБД с открытым исходным кодом *PostgreSQL* [6], в которой хранятся все данные системы MedMining. Асинхронная распределенная очередь задача *Celery* [7] позволяет выполнять ресурсоемкие задачи в фоновом режиме, получать информацию о ходе выполнения задач и др. Библиотека параллельных алгоритмов интеллектуального анализа данных *LibDM*, реализующих решения задач кластеризации, классификации и поиска ассоциативных правил.

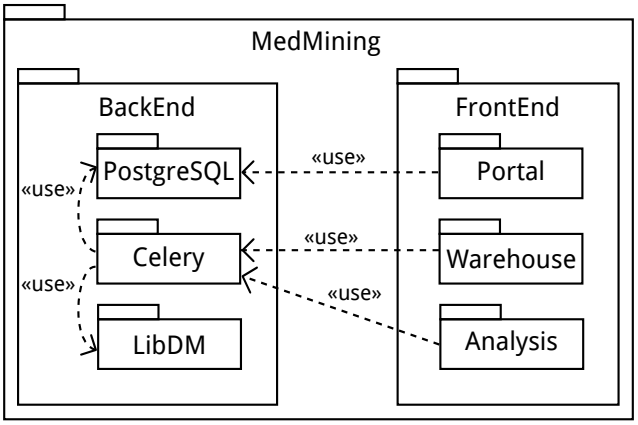

Рис. 4. Модульная структура системы MedMining

Интерфейс FrontEnd включает в себя следующие подсистемы. *Portal* представляет собой веб-приложение, написанное на основе фреймворка Django [8] и является тонким клиентом системы MedMining. Подсистема *Warehouse* отвечает за импорт данных физиологических исследований в хранилище на основе СУБД *PostgreSQL*. Функции импорта и преобразования подсистемы *Warehouse* работают асинхронно в виде заданий *Celery*. Подсистема *Analysis* предназначена для постановки задач интеллектуального анализа и визуализации результатов анализа.

3. Заключение

В работе представлена архитектура системы MedMining, которая предназначена для интеллектуального анализа данных физиологических исследований спортсменов. Система обеспечивает экспорт результатов измерений в хранилище данных. Поддерживается хранение как необработанных результатов измерений (значений, поступающих непосредственно с приборов), так и их обработанных аналогов (получаемых путем усреднения, аппроксимации или других интегрирующих действий над обработанными данными). Интеллектуальный анализ результатов измерений направлен на определение ключевых показателей результативности и эффективности методики тренировок и поиск трендов и аномалий в этих показателях.

Система позволяет Исследователю выполнять следующие основные функции:

- выработка заключения о состоянии конкретного спорстмена;
- проведение корректировок режима тренировок спортсмена;
- выявление наиболее успешной методики тренировок;
- исследование влияния тренировочного процесса на физиологическое состояние спортсмена (включая дополнительные параметры).

Исследования, представленные в данной работе, предполагается продолжить по следующим направлениям. Создание методов интеллектуального анализа данных, которые учитывают специфику данной предметной области (физиология профессиональных спортсменов). Другим направлением является применение параллельных СУБД [9, 13, 14] для обработки сверхбольших объемов данных физиологических исследований методами интеллектуального анализа данных [15].

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 12-07-00443-а.

Литература

1. Дюк, В.А. Предварительные результаты обработки разнотипных биометрических данных методами Data Mining / В.А. Дюк, О.В. Жвалевский, С.Б. Рудницкий, Д.А. Толстоногов // Труды СПИИРАН. – 2009. – № 9. – С. 197–210.

2. Han, J. Data Mining: Concepts and Techniques / J. Han, M. Kamber – Morgan Kaufmann, 2006. – 743 p.

3. Yoo, I. Data Mining in Healthcare and Biomedicine: A Survey of the Literature / I. Yoo, P. Alafaireet, M. Marinov, K. Pena-Hernandez, R. Gopidi, J.-F. Chang, L. Hua // Journal of Medical Systems. – 2012. – Vol. 36, No. 4. – P. 2431–2448.

4. Lo, B.P.L. Pervasive sensing for athletic training / B.P.L. Lo, A. Atallah, B. Crewther, et al. // Delivering London 2012: ICT Enabling the Games, The IET special interest publication. – 2011. – P. 53–62.

5. Московский, А.А. Исследование производительности суперкомпьютеров семейства «СКИФ Аврора» на индустриальных задачах / А.А. Московский, М.П. Перминов, Л.Б. Соколинский, В.В. Черепенников , А.В. Шамакина // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». – 2010. – № 35(211). – С. 66–78.

6. Stonebraker, M. The POSTGRES next-generation database management system / M. Stonebraker, G. Kemnitz // Communications of the ACM. – 1991. – Vol. 34, No. 10. – P. 78–92.

7. Celery: Distributed Task Queue // Celery Project. URL: `http://celeryproject.org` (дата обращения: 08.11.2012).

8. Ortiz, A. Web Development with Python and Django // Proceedings of the 43rd ACM technical symposium on Computer science education. ACM, – 2012. – P. 686.

9. Пан, К.С. Разработка параллельной СУБД на основе последовательной СУБД PostgreSQL с открытым исходным кодом / К.С. Пан, М.Л. Цымблер // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». – 2012. – № 18(277), Вып. 12. – С. 112–120.

10. Исаев, А.П. Функциональное состояние кардиореспираторной системы бегунов в первые два дня деакклиматизации после двадцати дней пребывания в верхнем среднегорье / А.П. Исаев, В.В. Эрлих // Вестник ЮУрГУ. Серия «Образование, здравоохранение, физическая культура». – 2012. – № 8 (267). – С. 34–37.

11. Исаев, А.П. Стратегии формирования адаптационных реакций у спортсменов. Основы теории адаптации и закономерности ее формирования в спорте высоких и высших достижений / А.П. Исаев, В.В. Рыбаков, В.В. Эрлих с соавт. // Вестник ЮУрГУ. Серия «Образование, здравоохранение, физическая культура». – 2012. – № 21 (280). – С. 46–56.

12. Эрлих, В.В. Ключевые морфометрические, спирографические и кардиопульмональные портретные характеристики ведущих спортсменов в возрасте 15-16 лет в состоянии покоя и оценка физической работоспособности в период участия в социально значимых соревнованиях / В.В. Эрлих, А.П. Исаев, В.В. Епишев, Ю.Б. Хусаинова // Вестник ЮУрГУ. Серия «Образование, здравоохранение, физическая культура». – 2011. – № 39 (256). – С. 19–21.

13. Соколинский, Л.Б. Обзор архитектур параллельных систем баз данных // Программирование. – 2004. – № 6. – С. 49–63.

14. Костенецкий, П.С. Технологии параллельных систем баз данных для иерархических многопроцессорных сред / П.С. Костенецкий, А.В. Лепихов, Л.Б. Соколинский // Автоматика и телемеханика. – 2007. – № 5. – С. 112–125.

15. Miniakhmetov, R.M. Integrating Fuzzy c-Means Clustering with PostgreSQL // Труды Института системного программирования РАН. – 2011. – Т. 21. – С. 263–276.

Виталий Викторович Епишев, кандидат биологических наук, доцент, кафедра теории и методики физической культуры и спорта, Южно-Уральский национальный исследовательский государственный университет (г. Челябинск, Российская Федерация), epishev74@mail.ru.

Александр Петрович Исаев, доктор биологических наук, профессор, кафедра теории и методики физической культуры и спорта, Южно-Уральский национальный исследовательский государственный университет (г. Челябинск, Российская Федерация), tmfcs@mail.ru.

Руслан Марсович Минахметов, аспирант, кафедра системного программирования, Южно-Уральский национальный исследовательский государственный университет (г. Челябинск, Российская Федерация), tavein@gmail.com.

Александр Вячеславович Мовчан, студент 3 курса, кафедра системного программирования, Южно-Уральский национальный исследовательский государственный университет (г. Челябинск, Российская Федерация), movchan174@gmail.com.

Алексей Сергеевич Смирнов, руководитель информационного отдела спортивного комплекса, Южно-Уральский национальный исследовательский государственный университет (г. Челябинск, Российская Федерация), 2231034@mail.ru.

Леонид Борисович Соколинский, доктор физико-математических наук, профессор, кафедра системного программирования, Южно-Уральский национальный исследовательский государственный университет (г. Челябинск, Российская Федерация), sokolinsky@acm.org.

Михаил Леонидович Цымблер, кандидат физико-математических наук, доцент, кафедра системного программирования, Южно-Уральский национальный исследовательский государственный университет (г. Челябинск, Российская Федерация), zymbler@gmail.com.

Вадим Викторович Эрлих, кандидат биологических наук, доцент, кафедра теории и методики физической культуры и спорта, Южно-Уральский национальный исследовательский государственный университет (г. Челябинск, Российская Федерация), tmfcs@mail.ru.

PHYSIOLOGICAL DATA MINING SYSTEM FOR ELITE SPORTS

V.V. Epishev, South Ural State University (Chelyabinsk, Russian Federation),
A.P. Isaev, South Ural State University (Chelyabinsk, Russian Federation),
R.M. Miniakhmetov, South Ural State University (Chelyabinsk, Russian Federation),
A.V. Movchan, South Ural State University (Chelyabinsk, Russian Federation),
A.S. Smirnov, South Ural State University (Chelyabinsk, Russian Federation),
L.B. Sokolinsky, South Ural State University (Chelyabinsk, Russian Federation),
M.L. Zymbler, South Ural State University (Chelyabinsk, Russian Federation),
V.V. Ehrlich, South Ural State University (Chelyabinsk, Russian Federation)

The paper presents the architecture of MedMining system, which is designed for mining physiological studies of sportsmen. The system provides export of measurements to data warehouse. Storing of raw measurements (coming directly from the devices) and their processed analogues (obtained by averaging, approximation or other integrating operations on the processed data) is supported. Mining of measurements aims to discovering of key performance indicators and performance training techniques, as well as to searching trends and anomalies in these indicators for a flexible changes of training schedule.

Keywords: data mining, data warehouse, physiological research, elite sports.

References

1. Djuk V.A., Zhvalevskij O.V., Rudnickij S.B., Tolstonogov D.A. Predvaritel'nye rezul'taty obrabotki raznotipnyh biometricheskih dannyh metodami Data Mining [Preliminary Results of the Processing of Different Types of Biometric Data by Means of Data Mining Methods] // Trudy SPIIRAN [SPIIRAN Proceedings]. 2009. No 9. P. 197–210.
2. Han J., Kamber M. Data Mining: Concepts and Techniques. Morgan Kaufmann, 2006. 743 p.
3. Yoo I., Alafaireet P., Marinov M., Pena-Hernandez K., Gopidi R., Chang J.-F., Hua L. Data Mining in Healthcare and Biomedicine: A Survey of the Literature // Journal of Medical Systems. 2012. Vol. 36, No. 4. P. 2431–2448.
4. Lo B.P.L., Atallah A., Crewther B., et al. Pervasive Sensing for Athletic Training. Delivering London 2012: ICT Enabling the Games, The IET special interest publication. 2011. P. 53–62.

5. Moskovskij A.A., Perminov M.P., Sokolinskij L.B., Cherepennikov V.V., Shamakina A.V. Issledovanie proizvoditel'nosti superkomp'juterov semejstva «SKIF Avrora» na industrial'nyh zadachah [Perfomance Analysis of the Supercomputer Family «SKIF Avrora» on Industrial Tasks] // Vestnik JuUrGU. Serija «Matematicheskoe modelirovanie i programmirovanie» [Bulletin of South Ural State University. Series «Mathematical Modeling, Programming & Computer Software»]. 2010. No 35(211). P. 66–78.

6. Stonebraker M., Kemnitz G. The POSTGRES next-generation database management system // Communications of the ACM. Oct. 1991. Vol. 34, No. 10. P. 78–92.

7. Celery: Distributed Task Queue // Celery Project. URL: `http://celeryproject.org` (дата обращения: 08.11.2012).

8. Ortiz A. Web Development with Python and Django // Proceedings of the 43rd ACM technical symposium on Computer science education. ACM, 2012. P. 686.

9. Pan C.S., Zymbler M.L. Razrabotka parallel'noj SUBD na osnove posledovatel'noj SUBD PostgreSQL s otkrytym ishodnym kodom [Development of a Parallel Database Management System on the Basis of Open-Source PostgreSQL DBMS] // Vestnik JuUrGU. Serija «Matematicheskoe modelirovanie i programmirovanie» [Bulletin of South Ural State University. Series «Mathematical Modeling, Programming & Computer Software»]. 2012. No 18(277), Iss. 12. P. 112–120.

10. Isaev A.P., Ehrlich V.V. Funkcional'noe sostojanie kardiorespiratornoj sistemy begunov v pervye dva dnja deakklimatizacii posle dvadcati dnej prebyvanija v verhnem srednegor'e [Functional State of Athletes' Cardiorespiratory System in Two First Days of Deacclimatization after Twenty Days in Higher Uplands] // Vestnik JuUrGU. Serija «Obrazovanie, zdravoohranenie, fizicheskaja kul'tura» [Bulletin of South Ural State University. Series «Education, Healthcare, Physical Culture»]. 2012. No 8 (267). P. 34–37.

11. Isaev A.P., Rybakov V.V., Ehrlich V.V. et al. Strategii formirovanija adaptacionnyh reakcij u sportsmenov. Osnovy teorii adaptacii i zakonomernosti ee formirovanija v sporte vysokih i vysshih dostizhenij [Strategy for the Formation of Adaptive Reactions of Athletes. Adaptation Theory Basics and Its Formation in the Elite Sport] // Vestnik JuUrGU. Serija «Obrazovanie, zdravoohranenie, fizicheskaja kul'tura» [Bulletin of South Ural State University. Series «Education, Healthcare, Physical Culture»]. 2012. No 21 (280). P. 46–56.

12. Ehrlich V.V., Isaev A.P., Epishev V.V., Husainova Ju.B. Kljuchevye morfometricheskie, spirograficheskie i kardiopul'monal'nye portretnye harakteristiki vedushhih sportsmenov v vozraste 15-16 let v sostojanii pokoja i ocenka fizicheskoj rabotosposobnosti v period uchastija v social'no znachimyh sorevnovanijah [Key Morphometric, and Cardiopulmonary Spirographic Portrait Characteristics for Leading Athletes in Age 15-16 at Rest and Evaluation of Physical Performance During Participation in Socially Important Events] // Vestnik JuUrGU. Serija «Obrazovanie, zdravoohranenie, fizicheskaja kul'tura» [Bulletin of South Ural State University. Series «Education, Healthcare, Physical Culture»]. 2011. No 39 (256). P. 19–21.

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

14. Kostenetskii P.S., Lepikhov A.V., Sokolinskii L.B. Technologies of parallel database systems for hierarchical multiprocessor environments // Automation and Remote Control. 2007. Vol. 68, No. 5. P. 847–859.

15. Miniakhmetov R.M. Integrating Fuzzy c-Means Clustering with PostgreSQL // Trudy Instituta sistemnogo programmirovanija RAN [Proceedings of ISP RAS]. 2011. Vol. 21. P. 263–276.

Поступила в редакцию 1 декабря 2012 г.

УДК 004.451.51

РЕАЛИЗАЦИЯ ТРАНСЛЯТОРА RAID-5 ДЛЯ РАСПРЕДЕЛЕННОЙ ФАЙЛОВОЙ СИСТЕМЫ GLUSTERFS[1]

А.С. Игумнов, А.Ю. Берсенев

Статья посвящена реализации алгоритма RAID-5 в распределенной файловой системе GlusterFS. Анализ требований предъявляемых к масштабируемой файловой системе (ФС), способной задействовать в дисковые ресурсы узлов вычислительного кластера, показывает, что реализация распределенной версии алгоритма RAID-5 позволяет существенно повысить устойчивость ФС к сбоям отдельных узлов и даже стоек кластера. В статье дается краткий обзор принципов функционирования распределенной файловой системы GlusterFS и описывается способ встраивания алгоритма RAID-5 в эту систему. Описываются основные алгоритмы и структуры данных, реализованные для адаптации RAID-5 в распределенную ФС. Делаются выводы об устойчивости и производительности разработанной ФС. Показано, что реализованный алгоритм позволяет наращивать пропускную способность ФС до пропускной способности нижележащей сетевой системы, незначительно теряя в производительности при наличии отказавших узлов.

Ключевые слова: распределенная файловая система, отказоустойчивость, RAID-5, GlusterFS.

Введение

Развитие вычислительных кластеров современного типа начиналось в 90-х годах прошлого века с объединения относительно недорогих персональных компьютеров с помощью ставшей в те годы коммерчески доступной сети 100 Мбит/с Ethernet. Для хранения данных в состав кластера включался дополнительный модуль, игравший роль файлового сервера, как правило, поддерживающего протокол NFS. Повышение производительности вычислительных узлов и увеличение их количества в составе единого суперкомпьютера сделали классический файловый сервер узким местом в архитектуре кластера.

Техническим решением этой проблемы стало увеличение пропускной способности каналов связывающих вычислительные узлы и файловый сервер (40 Гбит/с Infiniband или связки из двух-четырех каналов 1 Гбит/с Ethernet), а так же использование RAID для преодоления ограничения на пропускную способность единичного диска.

Алгоритмическим решением проблемы стало распараллеливание подсистемы ввода/вывода (В/В) — частичное (использование нескольких узлов В/В, подключенных к одному многоканальному RAID контроллеру), тотальное (использование множества относительно недорогих узлов В/В, оборудованных стандартными дисками), а также различные промежуточные варианты.

К сожалению, ни российский список TOP50 [1], ни международный TOP500 [2] не предоставляют информацию о подсистеме В/В суперкомпьютеров. На основе изучения публикаций можно сделать вывод, что компьютеры с максимальной производительностью, такие как кластер «Ломоносов» (Московский государственный университет имени

[1] Статья рекомендована к публикации программным комитетом Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: поиск новых решений – 2012».

М.В. Ломоносова) используют бездисковые вычислительные узлы и высокопроизводительные системы хранения, объединенные с помощью параллельной файловой системы (ФС) Lustre. В то же время, кластеры меньшего масштаба (кластер «Уран», Екатеринбург, Институт математики и механики УрО РАН) используют вычислительные узлы, снабженные локальными дисками, и высокопроизводительные RAID системы, доступные на узлах по NFS.

Данная статья посвящена разработке распределенной высокопроизводительной и надежной ФС, способной задействовать локальные диски узлов кластера. Разработанная система расширяет реализацию известной ФС с массовым параллелизмом — GlusterFS [3].

При разработке ФС ставились следующие задачи:

- возможность задействовать для хранения данных диски на узлах кластера, участвующих в вычислениях или выделенных;
- обеспечение целостности данных при выходе из строя единичного узла хранения или нескольких узлов из заранее известного домена отказа;
- масштабируемость пропускной способности системы В/В до скорости канала связи соединяющего узлы;
- возможность работы с большими наборами данных, существенно превышающих по объему емкость одного локального диска;
- минимальные накладные расходы на хранение служебной информации ФС.

Для реализации этих задач было принято решение реализовать в распределенной ФС алгоритм RAID-5, используя транспортный уровень и вспомогательные библиотеки ФС GlusterFS. Отдельной целью разработки стала оценка стабильности существующей версии GlusterFS и оценка гибкости, заложенных в нее механизмов расширения.

Поскольку в терминологии GlusterFS отдельные модули, расширяющие функциональность называются «трансляторами» в дальнейшем мы будем обозначать нашу ФС как RAID-5 Translator (R5T).

1. Описание структуры GlusterFS

GlusterFS – модульная распределенная сетевая ФС. Ее отличительной особенностью является отсутствие выделенного сервера метаданных, что позволяет осуществлять практически линейное масштабирование ФС на однородных узлах В/В. Первая версия GlusterFS вышла в 2006 году. В конце 2011 года GlusterFS была приобретена компанией Red Hat Inc. и в настоящий момент находится в стадии активной разработки и оптимизации.

И серверная и клиентская часть GlusterFS работают в пространстве пользователя через интерфейс FUSE (Filesystem in Userspace). Это облегчает разработку и отладку ФС, а так же позволяет защитить ядро ОС от ошибок в реализации драйверов ФС.

Ключевым элементом архитектуры GlusterFS являются так называемые «трансляторы» – программные модули, обрабатывающие и модифицирующие запросы на файловые операции. По своему функциональному назначению трансляторы делятся на следующие группы:

- трансляторы хранения данных, которые отвечают за взаимодействие с базовой ФС на узлах В/В;
- трансляторы передачи данных – сетевое взаимодействие;

- трансляторы, повышающие производительность – кэширование;
- трансляторы, добавляющие функциональность блокировки, права доступа;
- трансляторы кластеризации – наиболее важный класс трансляторов, реализующий алгоритмы распределения данных по узлам В/В.

Трансляторы в GlusterFS могут объединяться в сложную древовидную структуру. Когда GlusterFS получает вызов от файловой системы, он передает его транслятору, находящемуся в корне дерева трансляторов. Корневой транслятор, в свою очередь, передает вызов дальше подмножеству своих трансляторов-детей, те – своим детям и т.д. Результат вызова передается в обратном порядке – от листьев к корню, и затем – к приложению. На рис. 1 изображен пример объединения нескольких трансляторов, обеспечивающих локальное кэширование (io-cache, read-ahead), распределенную по сетевым узлам систему хранения файлов (unify) и взаимодействие с ФС на узлах В/В (POSIX).

2. Реализация транслятора RAID5 (R5T)

GlusterFS оперирует данными не на уровне отдельных блоков, а на уровне файлов. Для того, чтобы реализовать алгоритмы из RAID 5 каждый файл будем считать состоящим из блоков определенной длины. Блоки разных файлов независимы друг от друга. Размер блока задается при конфигурировании тома. По умолчанию он равен 128 КБ. Разбиение на блоки служит для распределения данных и контрольных сумм по узлам В/В и не накладывает на минимальный и максимальный размер операций В/В. Размещение блоков проще всего показать на примере RAID-5, собранного на трех узлах. Обозначим Дn – n-ный блок данных, а КС(i-k) — блок контрольной суммы, вычисленный на основе блоков Дi...Дk с помощью побайтовой операции XOR. Структура размещения фрагментов файлов будет следующая:

Первый узел: Д1, Д3, К(5-6)...
Второй узел: Д2, К(3-4), Д5...
Третий узел: К(1-2), Д4, Д6...

При выполнении операции чтения R5T вычисляет положение нужного блока на основе смещения искомого фрагмента от начала файла и отправляет через сетевой транслятор запрос на нужный узел. Контрольные суммы при операции чтения не используются.

При выполнении операции записи, R5T вычисляет положение нужного блока и положение соответствующей ему контрольной суммы, считывает их, производит вычисление новой контрольной суммы, выполняет две операции записи.

Операции работы с метаданными, такие как, chmod, mv и rm, будут выполняться на каждом узле В/В без изменений.

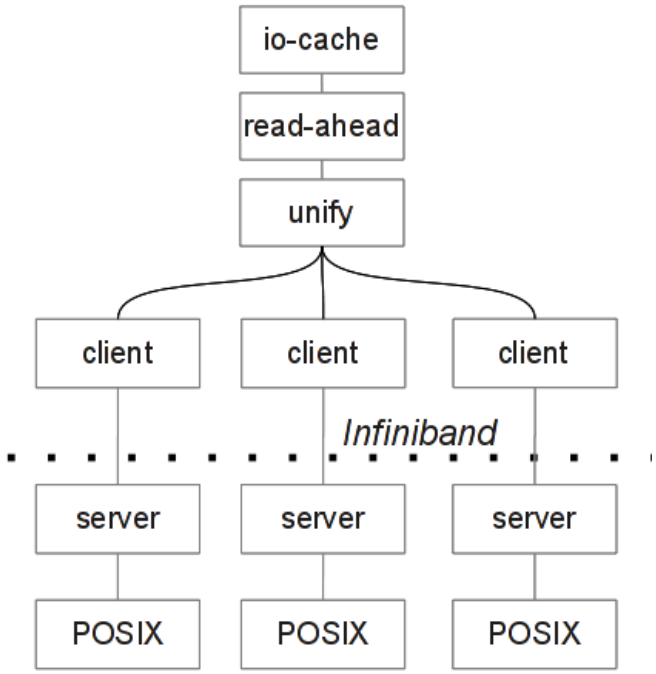

Рис. 1. Объединение трансляторов в GlusterFS

Хранение метаданных

В соответствии с идеологией GlusterFS все метаданные файла хранятся либо в метаданных фрагментов файла на узлах хранения, либо в расширенных атрибутах этих фрагментов. В случае с R5T все стандартные метаданные (uid, gid, права доступа и др.) хранятся в стандартных метаданных фрагментов файла, а в расширенных атрибутах хранятся метаданные, описывающие структуру RAID-5, а также метаданные, необходимые для восстановления после сбоя. Структура RAID-5 задается общим количеством узлов, а так же номером узла на котором размещается данный фрагмент файла. Для целей восстановления в расширенных атрибутах хранятся номер версии файла и номер версии метаданных файла. Если фрагмент файла восстанавливается после сбоя, то в его атрибутах дополнительно хранятся следующие величины: размер файла на момент начала восстановления, смещение от начала файла уже восстановленной области, номер версии файла, до которой производится восстановление (см. раздел 3, подраздел «Восстановление после сбоев»).

Отдельно стоит упомянуть про хранение размера файла. С одной стороны, общий размер хранимых в R5T данных превышает фактический размер файла за счет добавления контрольных сумм. С другой стороны, фрагмент файла, размещаемый на одном узле, существенно меньше файла в целом. Тем не менее, размер каждого из фрагментов можно сделать в точности равным размеру файла за счет использования «разреженных» (sparce) файлов.

Основные алгоритмы транслятора

Рассмотрим алгоритм размещения данных по узлам на примере вновь созданного файла нулевой длинны, размещенного на томе R5T, состоящем из N узлов В/В.

При записи первого байта файла:
на узле У(1) действительно записывается один байт данных,
на узлах У(2)-У(N-1) выполняется вызов seek, смещающий позицию записи, без выделения реального дискового пространства;
на узле У(N) выполняется запись КС (0 xor У(1)).

После заполнения 1го блока хранения порядок выполнения операций меняется:
У(1) — seek; У(2) — запись; У(3)-У(N-1) — seek;
на узле У(N) выполняется запись КС (У(1) xor У(2)), после чего выполняется seek, для придания файлу нужного размера.

Таким образом, размер фрагментов файла на всех узлах становится равным текущему размеру файла.

Алгоритмы транслятора, реализующие основные операции над файлами представлены на рис. 2–5.

1. Проверить, что недоступно не более одного узла.
2. Скорректировать флаги создания файла:
2.1. Убрать флаг O_APPEND, поскольку при использовании этого флага нельзя перемещаться по файлу с помощью вызова seek, а это необходимо для записи контрольной суммы.
2.2. Если установлен O_WRONLY, изменить его на O_RDWR, поскольку для обновления контрольной суммы, необходимо иметь возможность читать файл.
3. Передать вызов создания файла первому доступному подтому, указывая дополнительные расширенные атрибуты при создании: stripe-size, stripe-count, stripe-index, real-size и bad-node-index.
4. Если операция создания файла на первом подтоме завершилась неудачно, то удалить созданный фрагмент файла (если он был создан). Если удачно – передать вызов остальным дочерним трансляторам.
5. Установить файловый контекст и вернуть агрегированный результат работы.

Рис. 2. Алгоритм создания файла

1. Проверить, что недоступно не более одного узла.
2. Скорректировать флаги открытия файла:
2.1. Убрать флаг O_APPEND, т.к. при использовании этого флага нельзя перемещаться по файлу с помощью вызова seek, а это необходимо для записи контрольной суммы.
2.2. Если установлен O_WRONLY, изменить его на O_RDWR, потому что для обновления контрольной суммы необходимо иметь возможность читать файл.
3. Передать вызов открытия файла доступным узлам В/В.
4. Установить файловый контекст и вернуть агрегированный результат работы.

Рис. 3. Алгоритм открытия файла

1. Проверить, что недоступно не более одного узла.
2. Получить файловый контекст.
3. Если в процессе какой-либо записи не был доступен один из узлов, а сейчас недоступен другой, то читать файл нельзя, вернуть ошибку.
4. Вычислить первый и последний блоки, которые необходимо прочитать.
5. От первого до последнего блока, исключая блоки с контрольной суммой, сформировать запросы на чтение данных на узлах В/В. Если узел, с которого пытаемся читать недоступен, сформировать запросы на чтение ко всем другим узлам, чтобы восстановить информацию, которая была на недоступном узле.

Рис. 4. Алгоритм чтения из файла

1. Проверить, что недоступно не более одного узла.
2. Прочитать данные, которые находятся на месте записываемых.
3. Вычислить значение XOR между прочитанными данными и записываемыми.
4. Для каждой группы блоков(группа блоков — группа, имеющая общий блок с контрольной суммой) сформировать и выполнить запрос на запись блоков с данными, затем прочитать данные из блоков с контрольной суммой и выполнить операцию XOR с соответствующей частью данных, полученными на шаге 3.
5. Записать блоки с контрольной суммой.
6. Обновить реальный размер файла и данные файлового контекста, вернуть агрегированный результат работы. Если один из узлов был недоступен при записи, внести отметку об этом в расширенные атрибута файла на каждом доступном узле.

Рис. 5. Алгоритм записи в файл

Домены отказов

Узлы В/В GlusterFS размещаются на различных узлах кластера. Иногда возникают ситуации, которые приводят к недоступности целой группы узлов, находящихся в одном домене отказа, например, узлов, находящихся в одном здании, в одной серверной стойке или в сдвоенных узлах, в которых ремонт одного сервера требует отключения второго. R5T позволяет учитывать эту особенность. Тома рекомендуется создавать из узлов, находящихся в разных доменах отказа. Затем эти тома можно объединить в один, используя транслятор cluster/unify. На рис. 6 изображена файловая система, которая сохраняет работоспособность даже в случае отключения всех узлов в одном домене отказа. В этом случае пользователю доступно 2/3 от суммарной емкости дисков на узлах.

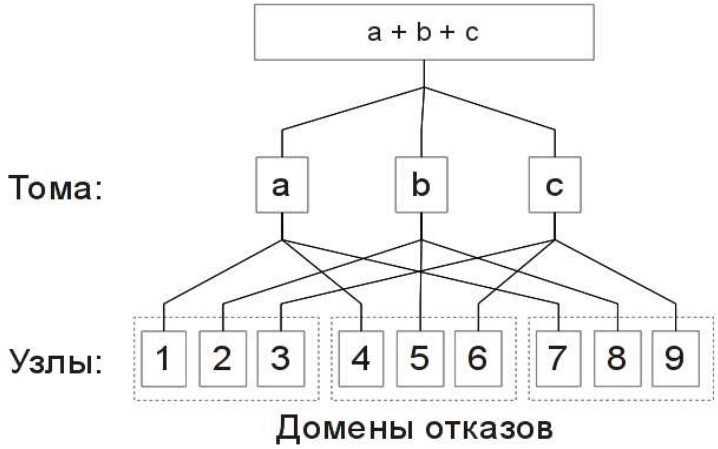

Рис. 6. Распределение узлов хранения по доменам отказов

3. Оценка производительности ФС

На операциях чтения транслятор R5T обеспечивает максимально возможную производительность. Такую же производительность показывает входящий в стандартную поставку GlusterFS транслятор cluster/stripe, но он не обеспечивает устойчивости к сбоям. Один клиент, читающий порциями меньше размера блока, будет ограничен пропускной способностью диска на отдельном узле. При параллельном чтении с нескольких клиентов, а так же при чтении большими порциями теоретическая пропускная способность R5T возрастает прямо пропорционально числу узлов В/В и ограничивается пропускной способностью использованной сети.

На операциях записи происходит двух-четырехкратное падение скорости по сравнению с теоретически возможным максимумом (достигается в трансляторе cluster/stripe). Падение, с одной стороны, определяется необходимостью выполнять операцию чтения перед каждой записью, а с другой стороны, при приближении к пределу пропускной способности сети, еще и необходимостью выполнения двух операции записи — данных и контрольной суммы. При увеличении количества узлов, теоретическая пропускная способность R5T растет и ограничивается сверху половиной пропускной способностью использованной сети.

Производительности ФС при сбоях

На операциях записи, а так же на операциях, манипулирующих метаданными, производительность ФС не изменяется. Все вычисления и обмены производятся в том же порядке и в тех же объемах, что и до появления сбоя кроме операции передачи данных на сбойный узел. Отсутствие передачи данных на сбойный узел может даже немного ускорить процесс записи, но при достаточно большом количестве узлов этот прирост станет практически незаметным.

Операция чтения замедлится за счет необходимости восстанавливать недостающие данные, хранящиеся на сбойном узле, по данным, хранящимся на остальных узлах В/В.

Рассмотрим набор из N узлов и операцию последовательного чтения блоками, соответствующими по размеру единичному блоку хранения. В такой ситуации $N-2$ операций чтения (пропускаем сбойный узел и контрольную сумму) пройдут в штатном режиме, а

N-ная операция потребует повторного считывания N−2 блоков, одного блока с контрольной суммой и вычисления результата XOR по всем полученным данным. Если учесть, что время выполнения XOR пренебрежимо мало по сравнению со скоростью передачи данных, то можно считать, что временные затраты на чтение N блоков будут равны 2N−3 единичных операций чтения. То есть, при достаточно большом числе узлов В/В мы всегда получаем фиксированное двукратное снижение производительности.

Данный алгоритм можно улучшить, разработав специализированный кэш, отслеживающий порядок чтения блоков. В этом случае, значение N-ого блока рассчитывается по мере считывания предыдущих N−1 блоков. Если данные из файла читались последовательно и файл не был изменен, то восстановление данных сбойного узла будет происходить практически без накладных расходов.

Существует «патологический» сценарий чтения из файла — небольшими фрагментами, размещенными исключительно на сбойном узле. В этом случае каждое считывание приводит к необходимости чтения данных со всех узлов, что может привести к падению скорости чтения прямо пропорциональному количеству задействованных узлов.

Восстановление данных после сбоя

С каждым фалом в R5T связан номер версии, который увеличивается при каждой записи в файл. В случае временной недоступности одного из узлов В/В и последующего его включения возможны две ситуации: файл не был изменен за время недоступности узла и его номер версии одинаков на всех узлах; номер версии на восстанавливаемом узле меньше, чем на остальных и, соответственно, данные хранящиеся на узле не соответствуют актуальному состоянию файла.

В первом случае узел В/В может быть задействован немедленно, без каких-либо вспомогательных операций. Во втором случае в фоновом режиме запускается процесс восстановления, заключающийся в считывании данных файла со всех узлов R5T, вычисление данных для восстанавливаемого узла и обновление их на диске.

Следует учесть, что запись в R5T после сбоя автоматически восстанавливает корректную структуру данных. Поэтому в процессе восстановления узел открывается для записи, но без обновления номера версии. В метаданные узла заносятся три величины: размер файла в момент начала восстановления, смещение от начала файла уже восстановленной области, номер версии файла, до которой производится восстановление. Первая величина позволяет обнаружить запись в конец файла новых — корректных данных, а две другие величины позволяют продолжить процедуру восстановления в случае повторного сбоя или перезапуска R5T. Каждая запись в файл обновляет номер версии, до которого идет восстановление. Если операция записи пересекается с уже восстановленной областью, то граница восстановленной области смещается до границы операции записи.

4. Оценка устойчивости GlusterFS

Несмотря на активную поддержку со стороны компании RedHat и довольно продолжительные сроки разработки, ФС GlusterFS содержит заметное количество ошибок и не до конца устоявшийся API.

В процессе тестирования ФС R5T, в базовых функциях GlusterFS была выявлена ошибка, которая, при определенном сочетании длинны запроса на запись и размера блока хранения, приводила к потере данных. Отчет об обнаруженной ошибке был отправлен

основной команде разработчиков. По результатам отчета в код GlusterFS были внесены изменения, устраняющие целый класс потенциальных ошибок. Было выявлено несколько утечек памяти. Исправляющий ошибки патч внесен в основную ветвь GlusterFS. Во время работы над R5T в API GlusterFS было внесено как минимум одно серьезное изменение, потребовавшее правки всего написанного кода, — добавлен новый параметр в целый ряд основных функций.

Несмотря на обнаруженные ошибки, активная работа, ведущаяся по совершенствованию GlusterFS фирмой RedHat, позволяет предположить, что эта ФС вскоре станет стабильной платформой, на которой смогут базироваться разнообразные распределенные ФС.

Заключение

Разработана масштабируемая файловая система с устойчивостью к одиночным сбоям узлов хранения данных. Показано, что на основе ФС Glusterfs возможно создание новых типов ФС со структурами данных, разнесенными по нескольким узлам хранения. Выявлена некоторая неустойчивость существующей версии ФС Glusterfs, которая ограничивает ее применение в полномасштабных промышленных приложениях.

Исходные тексты R5T доступны на сервере GitHub [4]

Работа выполнена в рамках программы Президиума РАН № 18 «Алгоритмы и математическое обеспечение для вычислительных систем сверхвысокой производительности» при поддержке УрО РАН (проект 12-П-1-1034).

Литература

1. Суперкомпьютеры. TOP50
   URL: http://top50.supercomputers.ru (дата обращения: 24.12.2012)
2. The Top500 List
   URL: http://www.top500.org (дата обращения: 24.12.2012)
3. Babu A. Gluster – «The GNU Cluster Distribution»
   URL: https://github.com/gluster/historic (дата обращения: 24.12.2012)
4. Исходные тексты R5T
   URL: https://github.com/alexbers/glusterfs_experiments/ (дата обращения: 24.12.2012)

Игумнов Александр Станиславович, Федеральное государственное бюджетное учреждение науки Институт математики и механики им. Н.Н.Красовского Уральского отделения Российской академии наук, igumnov@imm.uran.ru

Берсенев Александр Юрьевич, Федеральное государственное бюджетное учреждение науки Институт математики и механики им. Н.Н.Красовского Уральского отделения Российской академии наук, bay@hackerdom.ru

IMPLEMENTATION OF RAID-5 TRANSLATOR FOR GLUSTERFS DISTRIBUTED FILE SYSTEM

A.S. Igumnov, IMM UB RAS (Yekaterinburg, Russian Federation),
A.Yu. Bersenev, IMM UB RAS (Yekaterinburg, Russian Federation)

The article is devoted to the implementation of the RAID-5 algorithm as a part GlusterFS distributed file system. The article provides a brief overview of the principles of functioning of GlusterFS and describes how to embed RAID-5 into this file system. The article describes data structures and algorithms of RAID-5 translator. The conclusions of the stability and performance of translator are made. The article shows that implemented RAID-5 algorithm allows to achieve file system resiliency and to increase the throughput capacity of the file system. It is shown that damage of some storage nodes does not affect on write speed and slightly affect read speed of the translator.

Keywords: distributed file system, file system resiliency, RAID-5, GlusterFS.

References

1. Supercomputers. TOP50
   URL http://top50.supercomputers.ru (accessed: 24.12.2012)
2. The Top500 List
   URL http://www.top500.org (accessed: 24.12.2012)
3. Babu A. Gluster – «The GNU Cluster Distribution»
   URL https://github.com/gluster/historic (accessed: 24.12.2012)
4. R5T source code
   URL https://github.com/alexbers/glusterfs (accessed: 24.12.2012)

Поступила в редакцию 9 января 2013 г.

УДК 004.222

МОДУЛЯРНО-ПОЗИЦИОННЫЙ ФОРМАТ И ПРОГРАММНЫЙ ПАКЕТ ДЛЯ РАЗРЯДНО-ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ ВЫСОКОЙ ТОЧНОСТИ В ФОРМАТЕ С ПЛАВАЮЩЕЙ ТОЧКОЙ

К.С. Исупов

Рассматривается новый способ организации высокоточных вычислений с плавающей точкой, позволяющий распараллеливать арифметические операции вплоть до уровня отдельных цифр многоразрядных мантисс путем использования модулярно-позиционного формата представления данных. Основная концепция данного формата заключается в представлении мантисс чисел в многомодульной системе остаточных классов (СОК), а порядков – в позиционной системе счисления. Мантиссы сопровождаются позиционной характеристикой, которая способствует реализации эффективных алгоритмов выполнения немодульных операций в СОК, таких как деление (частный случай) и округление. На основе данного подхода разрабатывается программное решение High Precision Digit-Parallel Solver (HPDP-Solver). Комплекс HPDP-Solver может быть гибко настроен на конфигурацию конкретной машины, в результате чего обеспечивается наиболее эффективное использование ее ресурсов. В результате экспериментального исследования быстродействия пакета HPDP-Solver были получены результаты, доказывающие его преимущества при решении высокоточных численных задач перед имеющей мировую известность позиционной библиотекой GNU Multiple Precision Arithmetic Library. Пакет HPDP-Solver может быть применен при решении задач, которые предъявляют особо высокие требования к вычислительной точности.

Ключевые слова: плавающая точка, система остаточных классов, модулярно-позиционный формат, параллельная арифметика, высокоточные вычисления.

Введение

Проблема обеспечения достаточной точности вычислений всегда была актуальным направлением теоретических исследований в области компьютерной математики. Однако сейчас, как никогда ранее, несмотря на всю свою фундаментальную значимость, проблема точности приобретает все более четко выраженный прикладной аспект, проявляющийся, в первую очередь, при решении инженерных и научных задач большой размерности и высокой временной сложности, когда исследуемые процессы формализуются в виде дифференциальных уравнений, которые затем интегрируются численно, либо когда требуется обработка больших массивов исходных данных. Время решения таких задач даже при использовании суперкомпьютеров может составлять несколько часов, дней, месяцев. При таких «растянутых во времени» расчетах важна уверенность в получении корректных результатов, пригодных для практического применения. Причем, если алгоритмическую корректность решения можно верифицировать еще до начала вычислений, то задача определения значений погрешностей, возникающих на каждой итерации расчетов из-за недостаточной точности, является весьма нетривиальной проблемой.

Большинство численных методов оперируют над полем вещественных чисел, в силу этого актуальна проблема создания новых способов их компьютерной аппроксимации и организации параллельной обработки на многоядерных процессорах общего назначения, так как современные вычислительные системы реализуются в большинстве своем именно

на процессорах этого класса. В рамках данной проблемы приоритетными являются исследования, направленные на разработку новой математической базы, методов и масштабируемых алгоритмов высокоточных вычислений с плавающей точкой (ПТ), которые:

1) обеспечивают отображение всех величин, поддерживаемых используемыми в современных процессорах форматами с ПТ (наибольшее распространение получили двоичные форматы стандарта IEEE-754, включающие кроме чисел так же бесконечности и нечисловые величины – NaN), но лишены недостатков последних, связанных с ограничением точности: даже восьмикратной точности (4*double) зачастую оказывается недостаточно для корректного решения численной задачи, не говоря уже о двойной точности (double), поддерживаемой аппаратно производителями современных процессоров;

2) позволяют выполнять эффективное распараллеливание вычислений вплоть до уровня параллельной обработки отдельных разрядов мантисс операндов, так как не всегда алгоритмический параллелизм задачи позволяет при ее решении эффективно использовать все ресурсы многопроцессорной вычислительной системы;

3) обеспечивают минимизацию зависимости скорости выполнения арифметических операций от вычислительной точности этих операций.

В работе предлагается новый подход к организации высокоточных параллельных вычислений с плавающей точкой – модулярно-позиционный (квазимодулярный) формат данных и созданный на его основе программный пакет High Precision Digit-Parallel Solver (HPDP-Solver). Статья организована следующим образом. В разделе 1 дается формальное определение модулярно-позиционного формата, обосновывается его научная новизна, рассматриваются преимущества и области применения. Раздел 2 посвящен описанию основных модулей программного пакета HPDP-Solver. В разделе 3 приводятся результаты экспериментального исследования быстродействия пакета HPDP-Solver в сравнении с позиционной библиотекой высокой точности The GNU MP Bignum Library.

1. Модулярно-позиционный формат для высокоточных разрядно-параллельных вычислений с плавающей точкой

1.1. Формальное определение модулярно-позиционного формата

В двоичных форматах с плавающей точкой (здесь рассматриваются лишь форматы IEEE-754) любое вещественное число представляется трехэлементным набором:

$$[M,\ e,\ s], \tag{1}$$

где M принадлежит интервалу $[0, 2)$ и обозначает рациональную мантиссу, e принадлежит интервалу $[e_{\min}, e_{\max}]$ и выражает порядок (экспоненту), $e_{\min} = 2–2^{w-1}$, $e_{\max} = 2^{w-1}–1$, $s = \{0, 1\}$ – знак числа.

Величина чисел, записанных в таком формате, выражается формулой $–1^{e}*M*2^{e}$. Машинными представлениями чисел вида (1) являются $(w+t+1)$-разрядные двоичные векторы $(s\ r_w...r_2r_1d_t...d_2d_1)$, где разряды с d_1 по d_t отводятся под представление рациональных двоичных мантисс $M = d_t\,.\,d_{t-1}...d_2d_1$, разряды с r_1 по r_w отводятся под представление целочисленных порядков e, записанных в форме с избытком (в смещенной форме) $E = r_wr_{w-1}...r_2r_1 = e+e_{\max}$, разряд s выражает знак числа.

Принимая во внимание условие, что под целую часть рациональной мантиссы $M = d_t\,.\,d_{t-1}...d_2d_1$ в двоичных форматах вида (1) отводится 1 бит d_t, определим *целочисленную мантиссу* $M' = d_td_{t-1}...d_2d_1$ как t-разрядное неотрицательное целое двоичное число такое, что $M = M*2^{1-t}$. Определим *перемещенный порядок* λ, как целое двоичное число со

знаком такое, что $\lambda = 1–t+e$, , где e – w-разрядный порядок числа, представленного в двоичном формате (1).

Далее из множества $\{2^f+1, 2^f–1\}$ выберем n целочисленных положительных оснований (модулей) системы остаточных классов (СОК) [2, 3] $p_1,p_2,...,p_n$ таких, что для всех $i, j = 1,2,...,n$, при $i \neq j$: $\gcd(p_i, p_j) = 1$, где $\gcd(p_i, p_j)$ – наибольший общий делитель для p_i и p_j.

Затем целочисленную мантиссу $M' = d_t d_{t-1} \ldots d_2 d_1$ преобразуем в СОК, определяемую модулями $p_1,p_2,...,p_n$, получая тем самым *модулярную мантиссу* $M'' = (m_1,m_2,...,m_n)$:

$$M'' = (m_1,m_2,...,m_n) = (M' \bmod p_1,\ M' \bmod p_2,...,\ M' \bmod p_n),$$

где m_i принадлежит $[0,\ p_i–1]$ – i-ая знакопозиция (модулярный разряд) модулярной мантиссы M'', представляющая собой наименьший неотрицательный остаток от деления M' на i-ый модуль p_i.

Все операции над разрядами m_i модулярных мантисс $M'' = (m_1,m_2,...,m_n)$ будем выполнять относительно выбранных оснований $p_1,p_2,...,p_n$ согласно правилам выполнения арифметических операций в модулярной арифметике [2, 3]. Основания $p_1,p_2,...,p_n$ являются общими для всех операндов и хранятся в общей памяти вычислителей.

При всем этом с порядком λ продолжаем работать в двоичной системе и в СОК не преобразуем. Таким образом, любое число с плавающей точкой вида (1) можно представить в следующем *модулярно-позиционном (квазимодулярном) формате* (рис. 1):

$$[(m_1,m_2,...,m_n),\ \eta(M''),\ \lambda,\ s] \qquad (2)$$

где $M'' = (m_1,m_2,...,m_n)$ – модулярная мантисса числа, $\eta(M'')$ – функция, от модулярной мантиссы, значение которой отражает результат сравнения величины M'' и $(P–1)/2$, где $P = p_1*p_2*...*p_n$, $\lambda = 1–t+e$ – позиционный порядок, представляющий собой целое двоичное число со знаком, s – знак числа в модулярно-позиционном формате.

Значение функции $\eta(M'')$ определяется исходя из условий:

$$\eta(M'') = 1, \text{ если } M'' \leq (P–1)\ /\ 2,$$
$$\eta(M'') = –1, \text{ если } M'' > (P–1)\ /\ 2,$$
$$\eta(M'') = 0, \text{ если превосходство } M'' \text{ над } (P–1)\ /\ 2 \text{ не определено.}$$

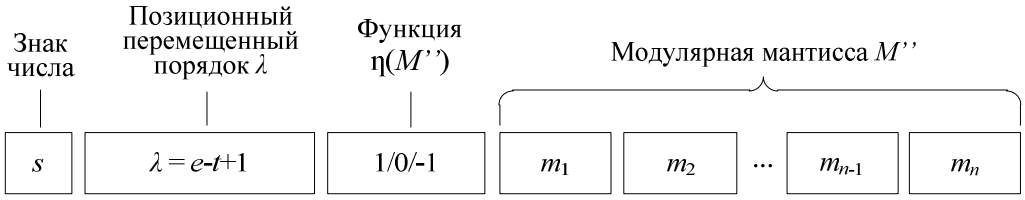

Рис. 1. Модулярно-позиционный формат с плавающей точкой

Сопровождение мантиссы каждого числа в модулярно-позиционном формате функцией $\eta(M'')$ позволит организовать эффективное выполнение алгоритмов работы со знаками, деления на двойку, минимизации числовой избыточности и округления модулярных мантисс. Все эти операции играют важную роль в организации вычислительного процесса над числами с плавающей точкой, и вместе с тем в системе остаточных классов реализуются весьма сложно. Для быстрого определения значения функции $\eta(M'')$ при известной модулярной мантиссе $M'' = (m_1,m_2,...,m_n)$ разработан специальный табличный алгоритм сравнения в СОК. Во время преобразования двоичной мантиссы M' к модулярной M'', определение значения функции $\eta(M'')$ будем выполнять на основании сопоставления M'

и (P–1) / 2, т.к. выполнять сравнение по величине позиционных чисел гораздо проще и быстрее, чем модулярных.

В формате (2) модулярные мантиссы $M'' = (m_1,m_2,...,m_n)$ в общем случае могут изменяться в пределах интервала [0, P-1], где $P = p_1*p_2*...*p_n$, поэтому становится возможным управлять точностью вычислений, задавая число и величину модулей p_i (как известно, точность в вычислениях с плавающей точкой определяется разрядностью мантисс операндов). Знакопозиции $m_1,m_2,...,m_n$ являются независимыми, что позволяет, при наличии многоядерных вычислителей, обрабатывать их параллельно и тем самым реализуется отдельный вид параллелизма - параллелизм на уровне арифметических операций.

Пример. Рассмотрим получение модулярно-позиционных представлений чисел на примерах. Пусть числа представляются в 10-разрядном двоичном формате вида (1), в котором под порядок e, отводится четыре бита (максимальный порядок $e_{\max} = 2^{4\text{-}1}–1$, соответственно $e = E–7$), под дробную часть мантиссы – пять бит (т.е. $t = 6$, причем целая часть d_6 рациональной мантиссы $M = d_6 . d_5 \ldots d_2 d_1$ в явном виде не записана) и под знак числа – один бит. Пусть для представления мантисс в модулярно-позиционном формате $[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$ выбраны три попарно взаимно простых модуля: $p_1 = 2^2–1$, $p_2 = 2^3–1$, $p_3 = 2^5–1$. Требуется перевести число $X = [1.5, –1, 0] = –1^0*1.5*2^{-1}$ из двоичного формата (1) в модулярно-позиционный формат (2).

С учетом принятых характеристик двоичного формата, число X будет записано в памяти ЭВМ в виде двоичного вектора (0 0110 10000).

1. Выделяем составные части числа X: знак числа $s = 0$, дробная часть рациональной мантиссы $d_5 \ldots d_2 d_1 = 10000_2$, порядок в форме с избытком $E = 0110_2 = 6$.

2. Восстанавливаем целую часть от M: $d_6 = 1$, т.к. $E > 0$, поэтому $M = 1.10000_2$.

3. Определяем порядок e: $e = E–e_{\max} = 6–7 = –1$, т.к. $E > 0$.

4. Определяем позиционный порядок λ и целочисленную мантиссу M':

$$\lambda = 1–t+e = –1–6+1 = –6,$$

$$M' = d_6 d_5 \ldots d_2 d_1 = 110000_2 = 48.$$

5. $M' <$ (P-1) / 2, следовательно $\eta(M'') = 1$.

6. Находим модулярную мантиссу $M'' = (m_1,m_2,m_3)$: $M'' =$ (48 mod 3, 48 mod 7, 48 mod 31) = (0,6,17).

В результате получили число X, представленное в модулярно-позиционном формате (2):

$$X = [(0,6,17), 1, –6, 0] = –1^0*(0,6,17)*2^{-6}.$$

Рассмотренный пример показывает, что в результате выбора трех оснований $p_1 = 2^2–1$, $p_2 = 2^3–1$, $p_3 = 2^5–1$, разрядность которых не превышает пяти бит, обеспечивается корректная работа с мантиссами, позиционная разрядность которых составляет десять бит ($P = p_1*p_2*p_3 = 651$), то есть точность аппроксимации увеличивается в два раза. При выборе большего числа оснований, диапазон допустимых значений модулярных мантисс может быть существенно расширен.

1.2. Обоснование новизны подхода

Необходимо отметить, что ранее, в работе [4], был представлен модулярный формат $[(a_1,a_2, ...,a_n), t]$, схожий с предлагаемым модулярно-позиционным форматом

$[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$. Однако модулярный формат $[(a_1,a_2, ...,a_n), t]$ служит для представления чисел с фиксированной, а не с плавающей точкой, и поэтому в нем модулярная величина $(a_1,a_2, ...,a_n)$ выражает разряды всего числа с учетом знака и порядка: знак числа $A = [(a_1,a_2, ...,a_n), t]$ определяется в соответствии с диапазоном, которому принадлежит модулярная мантисса $(a_1,a_2, ...,a_n)$, а порядок t принадлежит интервалу $[0, k_f]$, определяет позицию фиксированной точки в числе A и заложен в мантиссе $(a_1,a_2, ...,a_n)$, причем значение каждой знакопозиции a_i в формате $[(a_1,a_2, ...,a_n), t]$ определяется выражением $a_i = (K \,/\, 2^t) \bmod p_i$, где K – целое число такое, что $|K| \le 2^{n_f+k_f}-1$, а n_f и k_f – длины соответственно целой и дробной частей числа в позиционном формате с фиксированной точкой. Алгоритмы выполнения арифметических операций, а также округления и определения знака в модулярном формате с фиксированной точкой $[(a_1,a_2, ...,a_n), t]$, принципиально отличны от алгоритмов соответствующих операций над числами с плавающей точкой, для представления которых предлагается модулярно-позиционный формат $[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$.

Более того, модулярный формат $[(a_1,a_2, ...,a_n), t]$ не предусматривает сопровождение мантисс $(a_1,a_2, ...,a_n)$ их позиционными характеристиками для ускорения немодульных операций в СОК, такими, как функция $\eta(M'')$ в модулярно-позиционном формате.

Помимо модулярного формата, автор также предлагает единый модулярно-позиционный формат (МПФ) $[(a_1,a_2, ...,a_n), t, m_f, p_f]$, где m_f – мантисса числа во вложенном позиционном формате с плавающей точкой, p_f – порядок числа. Данный формат позволяет осуществлять обработку чисел так же и с плавающей точкой. Тем не менее, подход к использованию модулярных систем счисления в МПФ остается неизменным – в наборе $(a_1,a_2, ...,a_n)$ все также закодирован порядок и знак числа $A = [(a_1,a_2, ...,a_n), t]$ в формате с фиксированной точкой, а для работы с плавающей точкой используется позиционная мантисса m_f и порядок p_f.

Предлагаемый модулярно-позиционный формат с плавающей точкой предписывает рассмотрение модулярной мантиссы $M'' = (m_1,m_2,...,m_n)$ отдельно от порядка λ и знака s: значение каждого разряда m_i, вне зависимости от λ и s, определяется выражением $m_i = M' \bmod p_i$, где M' – целочисленная двоичная мантисса числа. Таким образом, при вычислении значения m_i не предполагается дополнительное деление M' на 2^{k_f}, если принимать за k_f длину дробной части исходной мантиссы. Поэтому $M'' = (m_1,m_2,...,m_n)$ не несет в себе информацию обо всем модулярно-позиционном числе, а лишь указывает на старшие значащие разряды в записи его модуля, без определения позиции разделяющей точки и знака. Для определения абсолютной величины числа $[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$ необходимо умножить M'' на 2^{λ}. Знак s – отдельный элемент в записи модулярно-позиционного числа.

Эти моменты определяют отличные от предложенных в работе [4] методы и алгоритмы записи чисел в модулярно-позиционном формате (формат $[(m_1,m_2,...,m_n), \eta(M''), \lambda, s]$ позволяет отображать все величины, в том числе и нечисловые, определенные стандартом IEEE-754), алгоритмы выполнения базовых арифметических операций, преобразования, округления модулярных мантисс, определения знака, и, как следствие, принципиально иной способ организации вычислений.

1.3. Преимущества модулярно-позиционного формата

Представление чисел с плавающей точкой в модулярно-позиционном формате (2) позволяет решить сразу несколько проблем, присущих аналогам:

1) Увеличивается точность вычислений, которая определяется разрядностью мантисс – для достижения точности 4*double достаточно выбрать восемь 32-битных модулей СОК.

2) Благодаря независимости знакопозиций m_i, обеспечивается возможность распараллеливания арифметических операций вплоть до уровня отдельных разрядов мантисс;

3) Решается задача минимизации зависимости времени выполнения операций от точности: при добавлении дополнительных оснований p_i для увеличения точности, в случае, если их число превышает число вычислительных устройств, время выполнения операций увеличивается линейно, но не экспоненциально или квадратично, как в высокоточных позиционных библиотеках.

5) Сопровождение каждой модулярной мантиссы M'' ее позиционной характеристикой $\eta(M'')$ позволяет, в отличие от формата, предложенного в работе [4], эффективно выполнять такие сложные для системы остаточных классов операции, как определение знака, округление, деление на двойку и пр.

6) В ЭВМ работа со значениями знакопозиций m_i будет осуществляться как с переменными скалярных арифметических типов, что позволит применять к ним высоко оптимизированные массовые операции, заложенные в известных библиотеках для организации параллельных вычислений (например, операцию REDUCTION библиотеки OpenMP, либо MPI_REDUCE интерфейса MPI). Применять подобные операции невозможно к числам, представленным в многоразрядных позиционных форматах.

1.4. Области применения модулярно-позиционного формата

Класс задач, при решении которых применение модулярно-позиционного формата может оказаться эффективным, достаточно широк и включает в себя: множество задач, сводимых к выполнению итеративных операций над массивами данных (например, сложные матричные операции над одним наборами исходных данных), задачи численного интегрирования дифференциальных уравнений (задачи подобного рода возникают при моделировании самых разнообразных физических явлений, например, при исследовании долговременной орбитальной эволюции небесных тел), задачи, решаемые с применением рекуррентных формул (например, вычисление суммы рядов), задачи решения СЛАУ, и другие задачи, решение которых составляет множество итераций, в результате чего накопление ошибок округления неизбежно приводит к потере основной доли значащих разрядов результата, и др. Другими словами, использование модулярно-позиционного формата видится целесообразным, когда время, затрачиваемое на прямое и обратное преобразование мало, по сравнению со временем расчетов.

На основе модулярно-позиционного формата разработаны эффективные параллельные алгоритмы выполнения операций (сложение, вычитание, умножение, выравнивание порядков и деление), алгоритмы округления модулярных мантисс (используются итерационные алгоритмы округления мантисс M'' до четного с последующим делением на двойку и увеличением порядков λ на единицу; число итераций округления определяется типом операции, которая должна быть выполнена в следующий момент времени), алгоритмы сравнения и др. В качестве примера на рис. 2 представлены схемы параллельных алгоритмов умножения и сложения модулярно-позиционных операндов.

Рис. 2. Примеры алгоритмов выполнения арифметических операций с плавающей точкой в модулярно-позиционном формате

Для реализации операций деления модулярных мантисс (общий случай), определения их выхода за допустимый диапазон представления и преобразования в позиционную систему, являющихся наиболее сложными операциями в СОК, предлагается использовать эффективные методы и алгоритмы, предложенные в работах [3, 5, 6].

Введение новых параллельных вычислительных структур в обязательном порядке должно сопровождаться проектированием схем декомпозиции данных по узлам и ядрам вычислительных устройств. Поэтому была произведена разработка и анализ схем распределения модулярно-позиционных структур данных между узлами и вычислительными ядрами в пределах узлов. На рис. 3 представлены упрощенные модели этих схем для систем с общей памятью. Для многопроцессорных систем данные схемы аналогичны.

2. Программный пакет High-Precision Digit-Parallel Solver

На основе модулярно-позиционного формата и созданного для него алгоритмического аппарата высокоточных разрядно-параллельных вычислений разрабатывается трехзвенный программный пакет *High-Precision Digit-Parallel Solver* (HPDP-Solver). Обобщенная структура пакета представлена на рис. 4.

Серверная часть является «ядром» пакета и реализует выполнение всех заложенных в нее алгоритмических решений по обработке модулярно-позиционных структур данных. Сервер состоит из трех частей: блок инициализации параметров, блок приведения данных к параллельным модулярно-позиционным структурам, вычислительный блок.

1. *Блок инициализации параметров* позволяет настроить программный пакет на решение конкретной задачи на конкретной вычислительной системе посредством задания доступных параметров вычислений. К таким параметрам относятся: число необходимых для использования вычислительных узлов, число вычислительных ядер в каждом узле, необходимость использования межпроцессорных коммуникаций, разрядность вычислительных устройств, число параллельных процессов, требуемая точность вычислений, схема распределения модулярно-позиционных структур данных.

2. *Блок приведения данных к параллельным модулярно-позиционным структурам* осуществляет взаимодействие с хранилищем, загружая исходные позиционные данные с плавающей точкой. Далее загруженные данные преобразуются к модулярно-позиционному формату и становятся доступными для вычислительного блока.

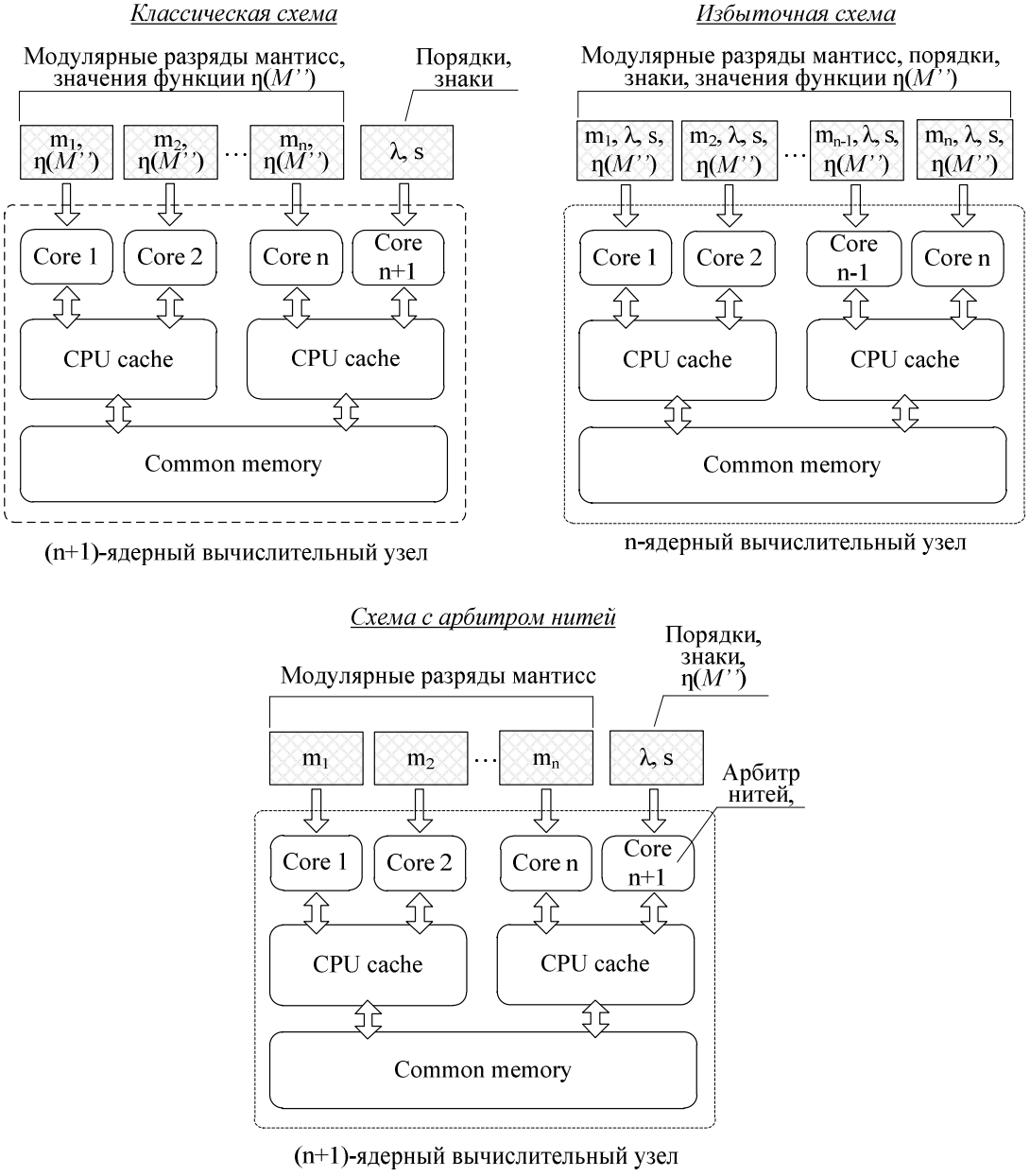

Рис. 3. Схемы распределения модулярно-позиционных структур данных для вычислительных систем с общей памятью

3. *Вычислительный блок* реализует выполнение разрядно-параллельных численных расчетов в модулярно-позиционном формате. В вычислительном блоке определяется полезный для конечного пользователя функционал пакета HPDP-Solver. К функциям, реализуемым вычислительным блоком, относятся:

3.1) Элементарные высокоточные разрядно-параллельные операции с ПТ: высокоточное умножение, высокоточное сложение, итерационное деление с заданной точностью, возведение в натуральную степень, вычисление логарифмов, вычисление прямых и производных тригонометрических функций.

3.2) Высокоточные операции над массивами данных (в настоящий момент ведется их разработка): вычисление скалярного произведения векторов, умножение матриц / векторов, сложение матриц / векторов, возведение матриц в степень, умножение матрицы / вектора на константу, поиск определителя и ранга матрицы, обращение матриц (предполагается использование итерационного алгоритма Гаусса с обратным ходом). Часть этих операций уже реализована и протестирована.

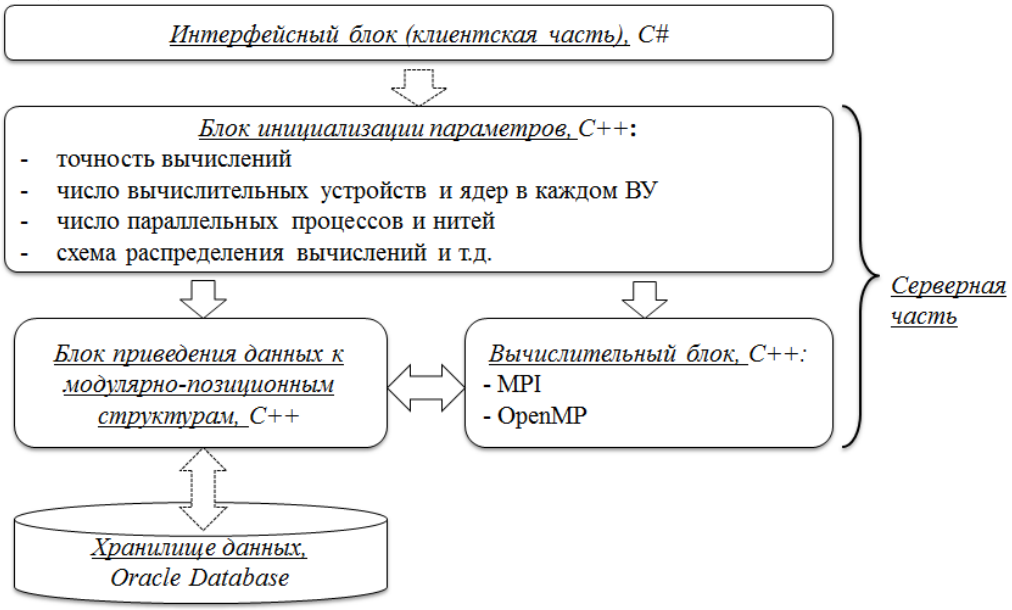

Рис. 4. Обобщенная структура программного пакета HPDP-Solver

В вычислительном блоке на программном уровне реализуются схемы распределения модулярно-позиционных структур данных, алгоритмы межпроцессорных обменов, а также базовые разрядно-параллельные алгоритмы выполнения высокоточных арифметических операций с плавающей точкой. *Основная концепция*, заложенная в вычислительном блоке – максимально-возможное отделение алгоритмического параллелизма задачи от арифметического параллелизма модулярно-позиционных структур, благодаря использованию гибридных технологий параллельной обработки (OpenMP+MPI).

В настоящее время производится разработка, отладка и тестирование основных функций для работы с массивами данных. Для каждой функции проектируется и исследуется специфическое алгоритмическое решение, отражающее природу модулярно-позиционных структур, нацеленное на параллельное исполнение и максимально возможную масштабируемость вычислительного процесса. Кроме этого, для каждой функции рассматриваются возможности ее программной реализации в различных спецификациях: MPI+OpenMP, MPI, OpenMP и CUDA.

3. Результаты экспериментов

Для оценки эффективности применения модулярно-позиционного формата при высокоточном решении численных задач над массивами данных с плавающей точкой, были проведены эксперименты, направленные на исследование быстродействия серверной части пакета HPDP-Solver в сравнении с широко известной во всем мире программной библиотекой The GNU MP Bignum Library (GMP [8]). *Целью эксперимента* являлось установление зависимости времени решения задачи от точности (т.е. от разрядности мантисс

операндов), при удалении в область больших и сверхбольших машинных диапазонов, и от числа используемых вычислительных ядер.

В качестве алгоритмической базы для эксперимента выступали высокоточные параллельные алгоритмы умножения плотных матриц $C = A*B$, $A,B,C \in R^{n\times n}$ и скалярного произведения векторов $c = a\times b$, $a,b,c \in R^n$. Данные операции, как база для исследований, выбраны не случайно: во-первых, они предполагают выполнение операций умножения с плавающей точкой, что приводит к увеличению разрядности мантисс, во-вторых, для получения каждого элемента результатного массива (либо результата скалярного произведения) необходимо сложить полученные частичные произведения, что позволяет оценить также скорость сложения и выравнивания порядков с плавающей точкой. Таким образом, оцениваются сразу три основные операции: умножение, сложение и выравнивание порядков.

При проведении численных экспериментов задействовалось от 8 до 32 вычислительных ядер кластерной системы Enigma (HP Hewlett-Packard Cluster Platform 3000 BL460c, Intel EM64T Xeon 5345, 2,3 ГГц, Infiniband) Вятского государственного университета. В качестве элементов матриц использовались положительные числа с плавающей точкой, имеющие разрядность мантиссы от 31 до 1891 бит и разрядность порядка 32 бита, т.е. изменяющиеся в пределах диапазона $[0, 2^{1891}*(2^{31}–1)]$. В качестве элементов векторов при вычислении скалярного произведения использовались положительные числа с плавающей точкой, имеющие разрядность мантиссы от 31 до 961 бит и разрядность порядка 32 бита.

Программный пакет HPDP-Solver выполнял умножение матриц согласно избыточной схеме распределения модулярно-позиционных структур, представленной на рис. 3 (т.е. потенциальный алгоритмический параллелизм задачи, реализуемый посредством разбиения одного из массивов на полосы либо блоки, не использовался). Достижение требуемой точности вычислений обеспечивалось путем увеличения числа оснований $p_1,p_2,...,p_n$, используемых для представления модулярных мантисс $M'' = (m_1,m_2,...,m_n)$, из расчета, что каждое основание, по величине не превосходящее 2^{31}, позволяет повысить точность на 31 бит. В программе, построенной на основе библиотеки GMP, была реализована классическая схема горизонтального ленточного разбиения.

На рис. 5 представлены графики зависимостей времени умножения матриц (размерности 1500×1500) от обеспечиваемой вычислительной точности при распараллеливании решения на 8, 16 и 32 вычислительных ядра. Легко заметить, что с увеличением разрядности элементов время решения задачи при использовании библиотеки GMP возрастает существенным образом. Так, при счете на восьми вычислительных ядрах и разрядности мантисс 31 бит задача решается за 431 секунду, а при разрядности мантисс 1891 бит – за 26361 секунд, т.е. более, чем за 7 часов! Таким образом, с увеличением вычислительной точности в 61 раз, время решения задачи линейно возросло также в 61 раз. При использовании пакета HPDP-Solver зависимость времени счета от точности гораздо менее выражена: при разрядности мантисс 31 бит задача была решена за 344 секунды, а при разрядности 1891 бит – за 2702 секунды, т.е. с увеличением точности вычислений в 61 раз, решение задачи замедлилось лишь в 7.8 раз. Таким образом, при работе с 1891-битными мантиссами пакет HPDP-Solver решил задачу в 9.7 раз быстрее, чем известная библиотека высокоточных вычислений GMP (при выполнении вычислений на восьми ядрах). Аналогичные зависимости наблюдаются при счете на 16 и 32 ядрах. Полученные результаты в полной мере соответствуют определенным априорным оценкам.

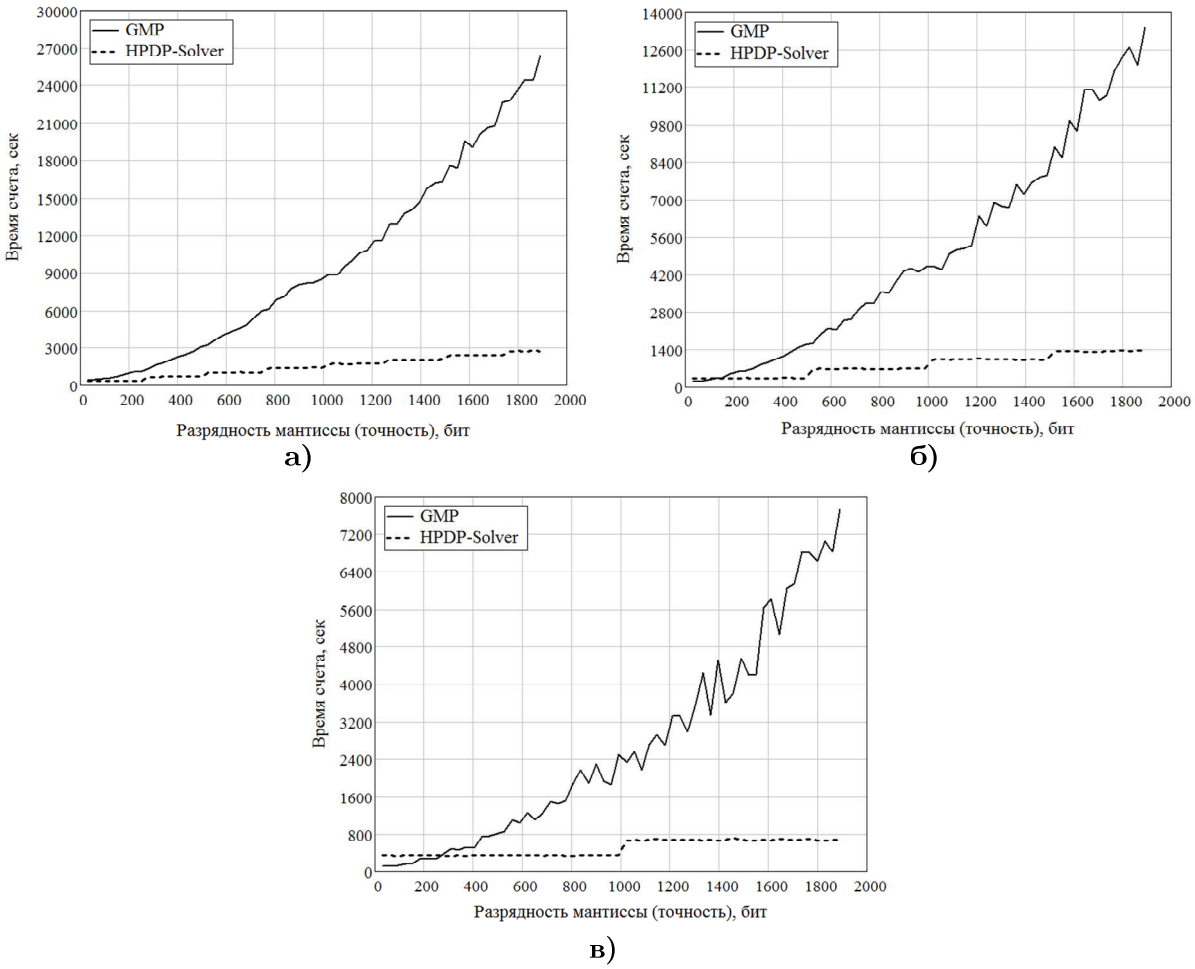

Рис. 5. Изменение времени умножения матриц при росте разрядности мантисс: а) счет выполнялся на 8 вычислительных ядрах, б) счет на 16 ядрах в) счет на 32 ядрах

Стоит заметить, что использование 16 ядер позволило вдвое сократить время счета как при использовании GMP, так HPDP-Solver, на всем диапазоне изменения разрядности мантисс. Аналогично, при счете на 32 ядрах время решения задачи сокращается примерно в 4 раза. Однако стоит учесть, что добиться таких высоких показателей масштабируемости решения на основе библиотеки GMP стало возможным лишь благодаря высокому алгоритмическому параллелизму самой задачи умножения матриц, т.к. арифметические операции в GMP не распараллеливались. Отсюда следует, что при решении высокоточных плохо распараллеливаемых на алгоритмическом уровне задач добиться линейного ускорения с использованием данной библиотеки будет невозможно.

На рис. 6 представлены сводные результаты исследования быстродействия программного пакета HPDP-Solver от точности вычислений при различном числе используемых параллельно работающих ядер. Анализ этих результатов позволяет сделать вывод, что эффективность и масштабируемость алгоритмов, заложенных в пакете HPDP-Solver, обуславливается лишь числом оснований для представления модулярно-позиционных структур и принципиально не зависит от решаемой задачи. Так, при счете на восьми ядрах время счета удваивается каждый раз, когда разрядность модулярных мантисс увеличивается на 248 бит: для корректной обработки модулярных мантисс $M'' = (m_1,m_2,...,m_n)$, позиционная разрядность которых больше, чем 248 бит, необходимо использовать больше

восьми 31-битных оснований p_1,p_2,...,p_n, поэтому минимум одно вычислительное ядро будет обрабатывать данные более чем по одному основанию p_i. Аналогично, для корректной обработки модулярных мантисс, имеющих разрядность более чем 526 бит, необходимо использовать более шестнадцати 31-битных оснований, и минимум одно вычислительное ядро будет обрабатывать данные более чем по двум основаниям. В свою очередь, при счете на 32 ядрах, для того, чтобы минимум одно вычислительное ядро обрабатывало данные более чем по одному модулярному разряду m_i, необходимо, чтобы позиционная разрядность модулярных мантисс была больше, чем 992 бита и т. д. Следовательно, вне зависимости от потенциального параллелизма задачи, пусть даже алгоритм ее решения будет строго последователен, характер зависимостей, представленных на рис. 6, останется неизменным.

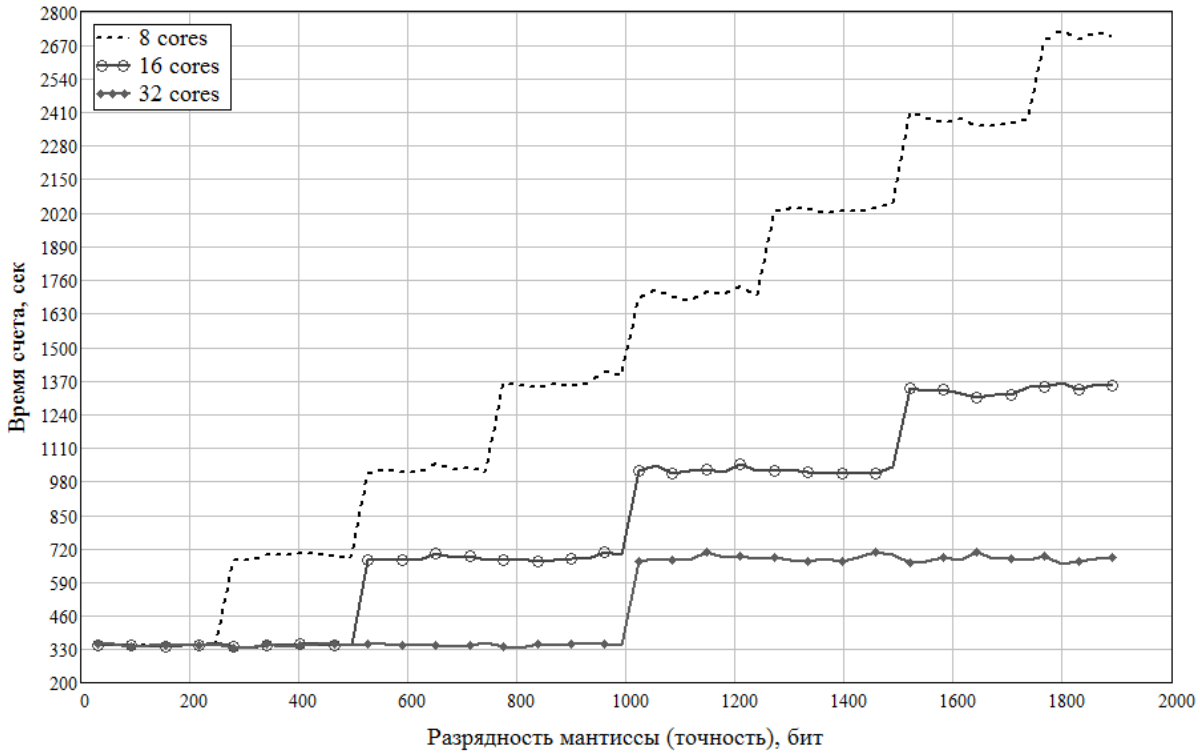

Рис. 6. Сводный график зависимостей быстродействия программного пакета HPDP-Solver от точности при решении задачи матричного умножения с использованием различного числа параллельно работающих вычислительных ядер

На рис. 7 представлен график зависимости времени выполнения операции скалярного произведения векторов, содержащих 1000000 элементов от точности при счете на восьми ядрах. При решении данной задачи параллельно вычислялись лишь частичные произведения, а их финальное суммирование было реализовано в последовательном режиме. Это объясняет рост времени счета от точности как в пакете HPDP-Solver, так и с использованием библиотеки GMP. Вместе с тем, характер зависимостей остается неизменным, и с увеличением точности счета наблюдается существенный отрыв по быстродействию пакета HPDP-Solver, относительно программного решения, построенного на основе GMP.

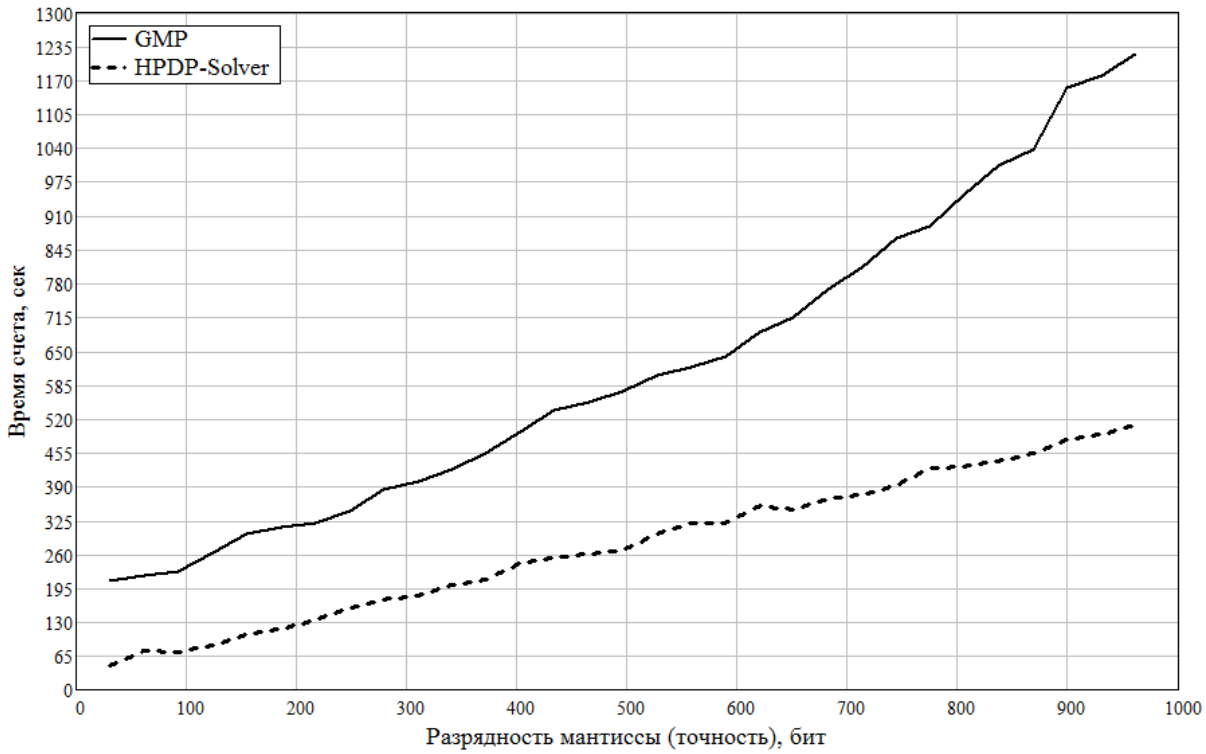

Рис. 7. Зависимость времени скалярного произведения векторов от точности при счете на 8 ядрах

Заключение

Предложен, обоснован и реализован новый способ организации высокоточных вычислений с плавающей точкой – модулярно-позиционный формат, особенностью которого является использование для представления мантисс чисел многомодульной системы остаточных классов, а для представления порядков – позиционной системы счисления, при этом величина числа выражается формулой $-1^{S}*M''*2^{\lambda}$, где $M'' = (m_1,m_2,...,m_n)$ – модулярная мантисса, сопровождаемая позиционной характеристикой $\eta(M'')$, выражающей соотношение между M'' и половиной произведения выбранных модулей, задающих СОК, λ – позиционный порядок, а s – знак числа. Это позволяет расширить диапазон представления операндов, повысить точность вычислений и распараллелить выполнение основных арифметических операций, сохраняя при этом скорость обработки порядков. Кроме этого, модулярно-позиционные структуры данных гораздо более компактны по сравнению со структурами, основанными на длинной позиционной арифметике, что позволяет значительно экономить память.

Для модулярно-позиционного формата разработаны эффективные параллельные алгоритмические решения, которые легли в основу серверной части разрабатываемого программного пакета HPDP-Solver. Относительная независимость основных частей пакета – клиента, сервера и хранилища данных, позволит применять его в самых различных областях науки и техники, где необходимо выполнять массивные высокоточные вычисления.

Проведенные эксперименты показали преимущества пакета HPDP-Solver перед имеющей мировую известность позиционной библиотекой GMP, позволив решить задачу высокоточного умножения матриц в 9,7 раз быстрее.

На данном этапе разработки (выполняется проектирование и реализация функций для работы с массивами данных) программный продукт HPDP-Solver удовлетворяет всем поставленным задачам:

1) Позволяет работать со всеми величинами машинных (IEEE-754) форматов данных с плавающей точкой, но обеспечивает значительно более высокую точность.

2) Позволяет минимизировать зависимость времени вычислений от точности.

3) Обеспечивает параллелизм вычислений на уровне арифметических операций.

4) Имеет возможности для адаптации как под конкретную задачу (посредством задания необходимой схемы распределения модулярно-позиционных структур данных), так и под конкретную высокопроизводительную вычислительную систему (посредством задания разрядности оснований для представления модулярных мантисс и их числа).

Пакет HPDP-Solver может быть применен при решении крупных задач, которые предъявляют особо высокие требования к вычислительной точности и сводятся к выполнению массовых арифметических операций с плавающей точкой, когда время, затрачиваемое преобразование данных, существенно меньше времени расчетов. Под эти условия попадает множество задач из самых различных областей науки и техники.

Статья рекомендована к публикации программным комитетом международной суперкомпьютерной конференции «Научный сервис в сети Интернет: поиск новых решений».

Литература

1. Исупов, К.С. Исследование эффективности современных средств поддержки высокоточных вычислений с вещественными числами / К.С. Исупов, А.Г. Иванов / Общество, наука, инновации (НТК-2012): Сб. материалов всероссийской научно-технической конференции (Киров, 16–27 апреля 2012 г.). – Киров: Изд-во ВятГУ, 2012. – 11 с.
2. Акушский, И.Я. Машинная арифметика в остаточных классах / И.Я. Акушский, Д.И. Юдицкий. – М.: Сов. Радио, 1968. – 440 с.
3. Omondi, A. Residue Number Systems: Theory and Implementation (Advances in Computer Science and Engineering Texts) / A. Omondi, B. Premkumar. – Imperial College Press, 2007. – 312 p.
4. Оцоков, Ш.А. Применение модулярной арифметики с фиксированной точкой для ослабления влияния ошибок округления компьютерных вычислений / Ш.А. Оцоков // Информационные технологии. – 2009. – № 12(160). – С. 50–54.
5. Sabbagh, A. New Arithmetic Residue to Binary Converters / A. Sabbagh, K. Navi // IJCSES International Journal of Computer Sciences and Engineering Systems. – 2007. – Vol. 1, No. 4. – P. 295–299.
6. Chang, C. A Division Algorithm For Residue Numbers / C. Chang, Y. Lai // Applied Mathematics and Computation. – 2006. – No. 172(1). – P. 368–378.
7. IEEE Standard for Floating-Point Arithmetic / IEEE. – NY 10016-5997. – USA. - 2008. – 70 p.
8. The GNU Multiple Precision Arithmetic Library. – URL: http://gmplib.org (дата обращения: 05.06.2011).

MODULAR-POSITIONAL DATA FORMAT AND SOFTWARE PACKAGE FOR DIGIT-PARALLEL HIGH-PRECISION FLOATING POINT CALCULATIONS

K.S. Isupov, *Vyatka State University (Kirov, Russian Federation)*

A new way of organization of high-precision floating point computations, which allows parallelizing arithmetic operations down to separate digits of multi-digit floating point mantissas through using a modular-positional data representation format, is considered. The main concept of this format is to represent the floating point mantissas in residue number system (RNS) and the exponent part in positional system. Floating point mantissas go with their positional characteristic that allows to successfully implement efficient algorithms for non-modular operations in RNS, such as division (special case), and rounding. Using this approach a software solution named High Precision Digit-Parallel Solver (HPDP-Solver) is developed. HPDP-Solver can be flexibly configured for a specific PC configuration, resulting in a more efficient use of its resources. The results obtained during the experimental performance study of HPDP-Solver proved its advantages in solving high-precision numerical problems if compared to a world-famous GNU Multiple Precision Arithmetic Library. HPDP-Solver can be used to solve problems that have some special demands on computational precision.

Keywords: floating point, residue number system, modular-positional data format, parallel arithmetic, high-precision calculations.

References

1. Isupov K.S., Ivanov A.G. Issledovanie jeffektivnosti sovremennyh sredstv podderzhki vysokotochnyh vychislenij s vewestvennymi chislami [Research Effectiveness Of Modern Means For High-Precision Calculations with Real Numbers]. Obwestvo nauka innovacii (NTK-2012): Sb. materialov vserossijskoj nauchno-tehnicheskoj konferencii (Kirov, 16-27 aprelja 2012) [Society, Science, Innovation (STC-2012): Sat. Materials of the Russian Scientific and Technologic (Kirov, Russia, April, 16-27, 2012)]. Kirov: Publishing of the Vyatka State University, 2012. 11p.
2. Akushskii I.Y., Yuditskii D.I. Mashinnaja arifmetika v ostatochnyh klassah [Computer Arithmetic in Residual Classes]. Sov. Radio, 1968. – 440 p.
3. Omondi A., Premkumar B. Residue Number Systems: Theory and Implementation (Advances in Computer Science and Engineering Texts. Imperial College Press, 2007. 312 p.
4. Ocokov Sh.A. Primenenie moduljapnoj apifmetiki s fiksipovannoj tochkoj dlja oslablenija vlijanija oshibok okpuglenija komp'jutepnyh vychislenij [Application of Fixed Point Modular Arithmetic for Reducing Influence of Round Errors of computer computation]. Informacionnye tehnologii [Information Technology]. 2009. No. 12(160). P. 50–54.
5. Sabbagh A., Navi K. New Arithmetic Residue to Binary. IJCSES International Journal of Computer Sciences and Engineering Systems. 2007. Vol. 1, No. 4. P. 295–299.
6. Chang C., Lai Y. A Division Algorithm For Residue Numbers. Applied Mathematics and Computation. 2006. No. 172(1). P. 368–378.
7. IEEE Standard for Floating-Point Arithmetic. IEEE. NY 10016-5997. USA. 2008. 70 p.
8. The GNU Multiple Precision Arithmetic Library. URL: http://gmplib.org (accessed: 05.06.2011)

Поступила в редакцию 10 января 2013 г.

УДК 519.687

ПАРАЛЛЕЛЬНАЯ РЕАЛИЗАЦИЯ МЕЛКОЗЕРНИСТЫХ АЛГОРИТМОВ В СИСТЕМЕ WINALT[1]

М.Б. Остапкевич

Дано краткое описание системы имитационного моделирования алгоритмов и структур с мелкозернистым параллелизмом WinALT. Отличительные черты системы – визуальное построение и отладка моделей, а также ориентация не только на клеточный автомат и некоторые его расширения, но и на широкий спектр других мелкозернистых алгоритмов. Рассмотрена существующая подсистема параллельного исполнения, которая позволяет выполнять моделирование с использованием кластеров Windows машин. Сформулированы требования к новой проектируемой подсистеме параллельного исполнения, которая пригодна для исполнения моделей на широком спектре современных параллельных ЭВМ. Предложена ее архитектура, рассмотрены режимы параллельного исполнения моделей и сформулированы планы развития системы.

Ключевые слова: мелкозернистый параллелизм, имитационное моделирование, параллельное программирование, система моделирования, алгоритм параллельных подстановок.

Введение

Мелкозернистый параллелизм (МЗП), наряду с крупноблочным, является одним из двух основных видов параллелизма. Его отличительная черта — огромное число простых параллельно работающих процессорных элементов. К широко известным классам алгоритмов с мелкозернистым параллелизмом относятся классический клеточный автомат (КА) и его расширения (КА с окрестностью Марголуса [1], асинхронный КА, вероятностный КА), сеть КА, клеточно-нейронная сеть, дискретная динамическая сеть, системы Линденмайера. Алгоритмы и структуры с МЗП используются для моделирования физических [1], химических, биологических и социальных процессов, разнообразных вычислительных устройств (ассоциативный процессор, систолическая структура, однородная среда, универсальная вычислительная среда, конвейеры арифметических устройств).

Объем данных и сложность правил их трансформации в МЗП моделях делает практически невозможным их исследование без системы моделирования. Известно несколько десятков систем моделирования МЗП [2, 3]. Их общий недостаток — работа только с классическим КА и некоторыми его расширениями. Для моделирования всего спектра МЗП алгоритмов была построена среда моделирования WinALT [4]. Цель этого проекта — разработка системных алгоритмов и реализация системы моделирования WinALT на широком спектре доступных параллельных вычислительных средств. Такая реализация обеспечит возможность работы с большими объемами данных и вычислений, характерными для большинства практически полезных моделей.

[1] Статья рекомендована к публикации программным комитетом Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: поиск новых решений – 2012».

В статье дана общая характеристика среды моделирования WinALT, ее текущее состояние, описана существующая подсистема параллельного исполнения WinALT (ParSIM/Win32) и сформулированы требования к проектируемой подсистеме параллельного исполнения (ParSIM/Omni), предложена ее архитектура и сформулированы планы развития системы.

1. Общая характеристика среды моделирования WinALT

Среда моделирования WinALT реализует набор пользовательских функций, позволяющих создавать, редактировать, отлаживать и исполнять модели МЗП алгоритмов. Среда предоставляет возможности как визуального проектирования моделей, так и построения их аналитических описаний. Визуальный подход позволяет представлять графически не только данные моделей, но и правила их трансформации. Аналитический подход позволяет описать любые аспекты работы моделей (как содержательные, так и технические) на языке моделирования системы.

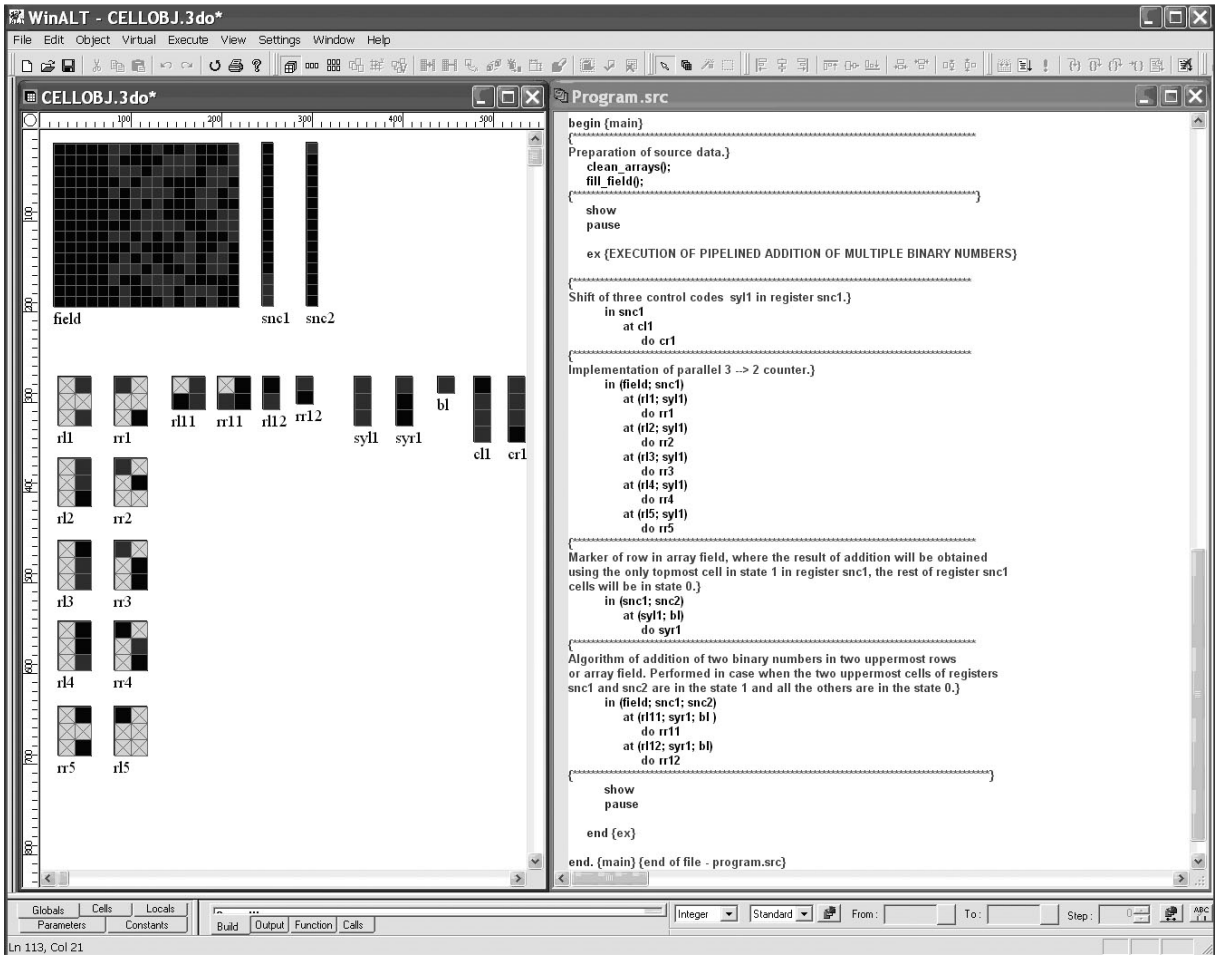

Рис. 1. Снимок экрана с примером модели сумматора многих чисел 3 в 2

Модель на рис. 1 имитирует работу сумматора многих чисел. В левой части окна показаны объекты данных (клеточные массивы и шаблоны подстановок) модели. До исполнения модели строки клеточного массива field содержат двоичные числа. После исполнения все строки клеточного массива field, кроме верхней, содержат нули. Верхняя строка содержит результат суммирования. Клеточные массивы snc1 и snc2 — статусные

регистры, показывающие, для каких строк выполнение сложения завершается. Все остальные объекты данных описывают левые и правые части подстановок (шаблоны подстановок). Левая часть определяет условие применимости подстановки, а правая — новое состояние клеток, изменяемых подстановкой. В правой части окна показана моделирующая программа сумматора. Она состоит из двух частей — сервисной и содержательной. Сервисная часть служит для заполнения обрабатываемых объектов начальными данными (вызов процедур clean_arrays и fill_field). Содержательная часть описывает необходимые подстановки и режим их применения. Оператор ex задает синхронный режим исполнения подстановок до их неприменимости. Оператор in указывает имя клеточного объекта (или имена нескольких клеточных объектов для более сложных подстановок), который будет трансформироваться описываемой подстановкой. Оператор at указывает шаблон (или шаблоны) левой части подстановок. Оператор do задает шаблон (или шаблоны) правой части подстановки.

Дистрибутив среды моделирования и коллекция примеров моделей доступны на [2]. Среда имеет открытую архитектуру, что позволяет пользователям добавлять в нее новые функции, форматы данных, режимы отображения, режимы моделирования и фрагменты моделей и формировать из них библиотеки.

В частности, библиотека форматов данных содержит поддержку форматов растровых изображений. На рис. 2 показана простейшая модель визуальной криптографии [5],

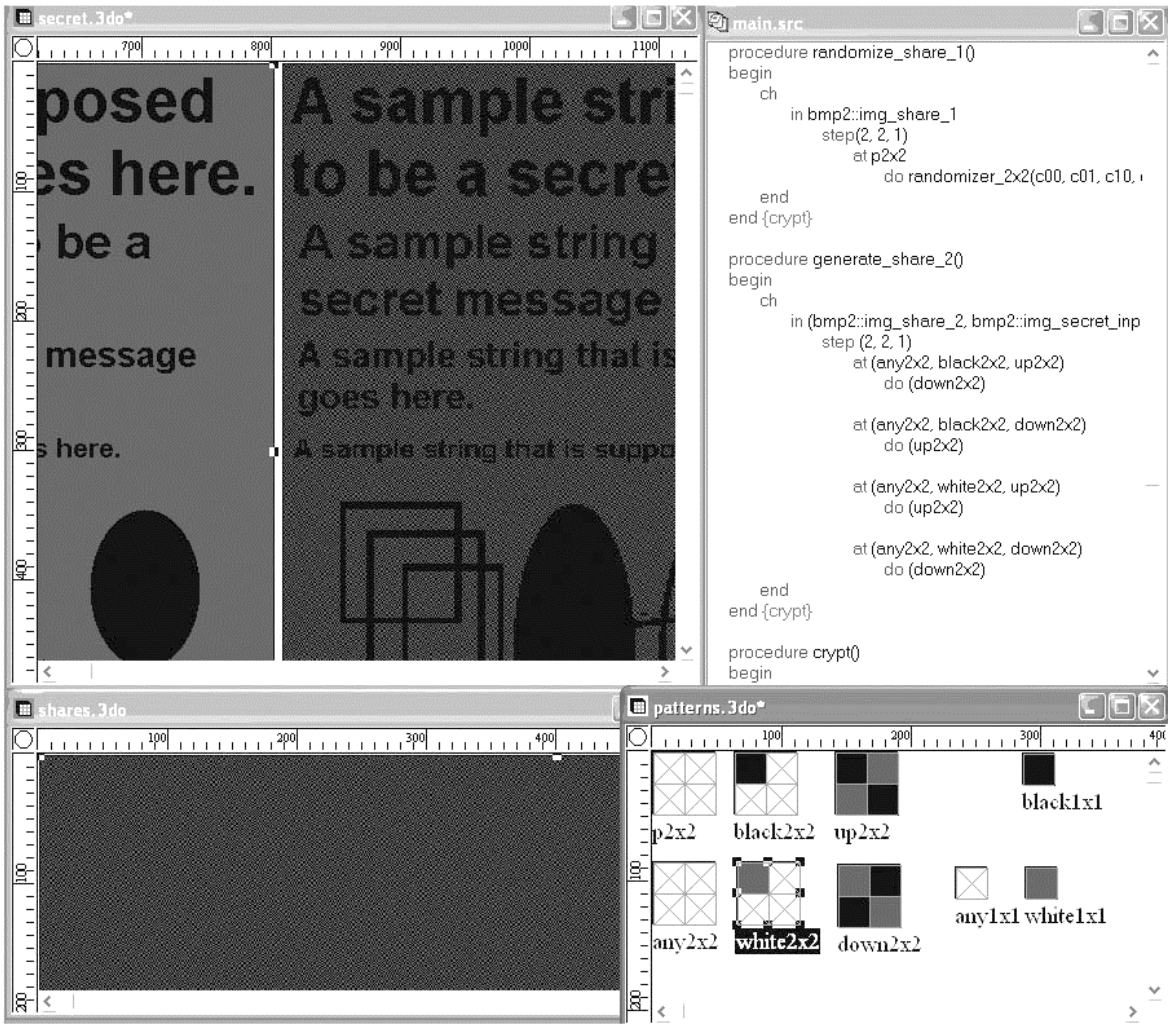

Рис. 2. Пример модели визуальной криптографии

которая использует их для хранения начальных, промежуточных и конечных данных. Левая часть окна secret.3do содержит исходное изображение, которое требуется зашифровать. Два изображения в окне shares.3do, одно из которых видно на рисунке, хранят изображение в зашифрованном виде. Изображение в правой части окна secret.3do содержит результат дешифрации.

Среда моделирования WinALT представлена следующими основными подсистемами:

1) консольная версия системы моделирования (исполняет модели без отладки, визуализации и взаимодействия с пользователем);

2) графическая среда WinALT (обеспечивает визуальное построение, отладку и исполнение моделей, имеет развитые средства взаимодействия с пользователем);

3) web среда (содержит дистрибутив системы, документацию, коллекцию моделей, массив информации по тематике МЗП);

4) подсистема параллельного исполнения ParSIM/Win32 (исполнение моделей на параллельных вычислительных системах).

ParSIM/Win32 реализована как приложение, исполняемое на невыделенном кластере компьютеров с 32 битной платформой Windows (Win32). Для передачи данных и координации работы был разработан собственный протокол. Его реализация базируется на Windows socket library. ParSIM/Win32 обеспечивает параллельное исполнение без каких-либо изменений в моделирующих программах для моделей, в которых все постановки имеют локальные окрестности 3x3 клетки; ParSIM/Win32 с одной стороны показала большую полезность для уменьшения времени моделирования, увеличения допустимых размеров объектов данных в моделях. С другой стороны, ее использование выявило недостатки, главные из которых:

1) реализация на платформе Windows, перенос которой на другие платформы трудоемок;

2) негибкая архитектура виртуальной машины (модуля, который управляет исполнением любой модели в системе), которая не допускает ее перенос на ряд современных архитектур (GPU) при сохранении достаточной эффективности;

3) поддержка распараллеливания только такого класса моделей, в которых все постановки имеют локальные окрестности 3x3 клетки;

4) привязка к единственному собственному протоколу передачи данных и координации работы удаленных процессов распределенного приложения.

С учетом указанных недостатков ParSIM/Win32 актуально построение новой подсистемы параллельного исполнения (ParSIM/Omni), преодолевающей их.

Главные требования к ParSIM/Omni следующие:

1) эффективное исполнение широкого спектра МЗП моделей на мультикомпьютерах в целом и на отдельных счетных узлах в частности;

2) использование существующей сетевой инфраструктуры и компьютеров с разнообразными конфигурациями, архитектурой и набором системного программного обеспечения без перенастройки WinALT;

3) поддержка различных протоколов доступа и обмена данными для взаимодействия с кластерами и между вычислительными устройствами,

4) возможность исполнения моделей, содержащих объекты данных, которые не могут быть размещены целиком в основной/виртуальной памяти.

2. Архитектура ParSIM/Omni

Предлагаемая архитектура, с одной стороны, призвана удовлетворить предъявленным требованиям, а с другой стороны в максимальной степени использовать уже существующие реализации модулей среды моделирования.

На верхнем уровне можно выделить следующие основные компоненты (рис. 3):

1) виртуальная машина;
2) менеджер объектов;
3) менеджер режимов моделирования;
4) менеджер протоколов и интерфейсов.

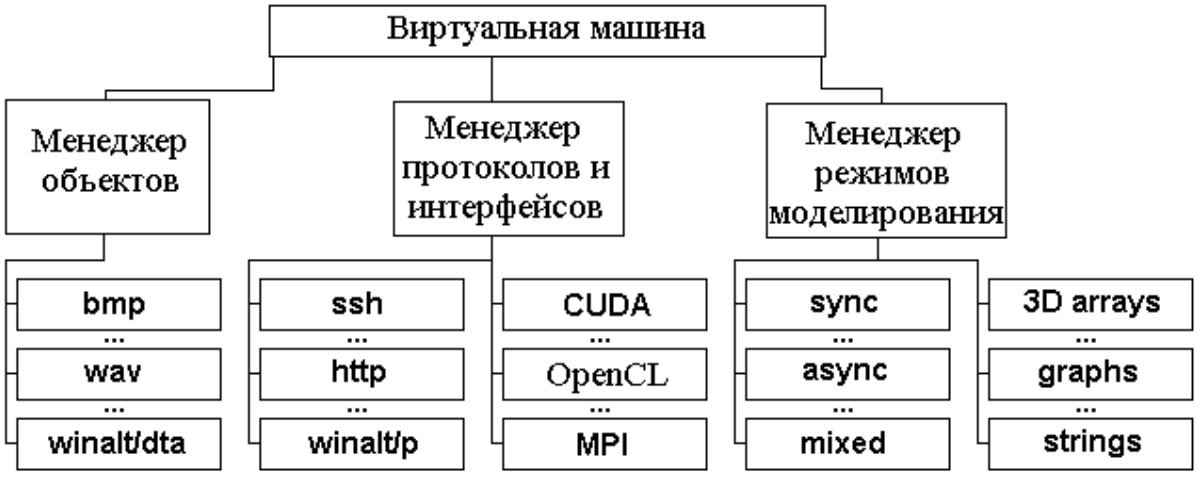

Рис. 3. Основные компоненты ParSIM/OMNI

Первые два из них имеются в существующей среде WinALT и, в частности, в ParSIM/Win32, но их свойства и набор функций претерпел изменения. Остальные являются новыми компонентами.

Виртуальная машина управляет исполнением моделирования, пользуясь операциями, реализованными во всех остальных основных компонентах. Новая версия виртуальной реализует не только операцию запуска модели на исполнение, но также и операции приостановки, возобновления и прекращения исполнения, посылки сигналов о событиях исполняемой модели. Кроме этого, новая версия позволяет одновременное исполнение нескольких процессов внутри одной виртуальной машины. Хотя набор инструкций виртуальной машины максимально приближен к уже имеющемуся, минимальный набор обязательных инструкций существенно сокращен, что делает ее реализацию на новых платформах, включая GPU, менее трудоемкой.

Менеджер объектов обеспечивает унифицированный доступ к клеточным объектам данных по имени клеточного объекта и координатам клетки вне зависимости от используемых форматов данных и протоколов доступа. Новыми свойствами менеджера являются: 1) возможность частичной загрузки объекта в виртуальную память (необходимо для работы со сверхбольшими объектами данных), 2) возможность загрузки и сохранения объектов, не являющихся файлами локальной файловой системы, а доступных по какому-либо протоколу.

Менеджер режимов моделирования реализует операции поиска и применения подстановок и обеспечивает следующие возможности пользователю:

1) выбирать более одного режима синхронности внутри одной модельной программы,

2) использовать в качестве объектов данных не только клеточные массивы, но и иные структуры, например, графы,

3) использовать разные режимы распараллеливания.

В ParSIM/Win32 есть только режим распараллеливания с разрезанием объектов данных и их распределением по счетным узлам (рис. 4). Он является примером параллелизма по данным и обеспечивает эффективное распараллеливание только для моделей с небольшими локальными окрестностями правил подстановки.

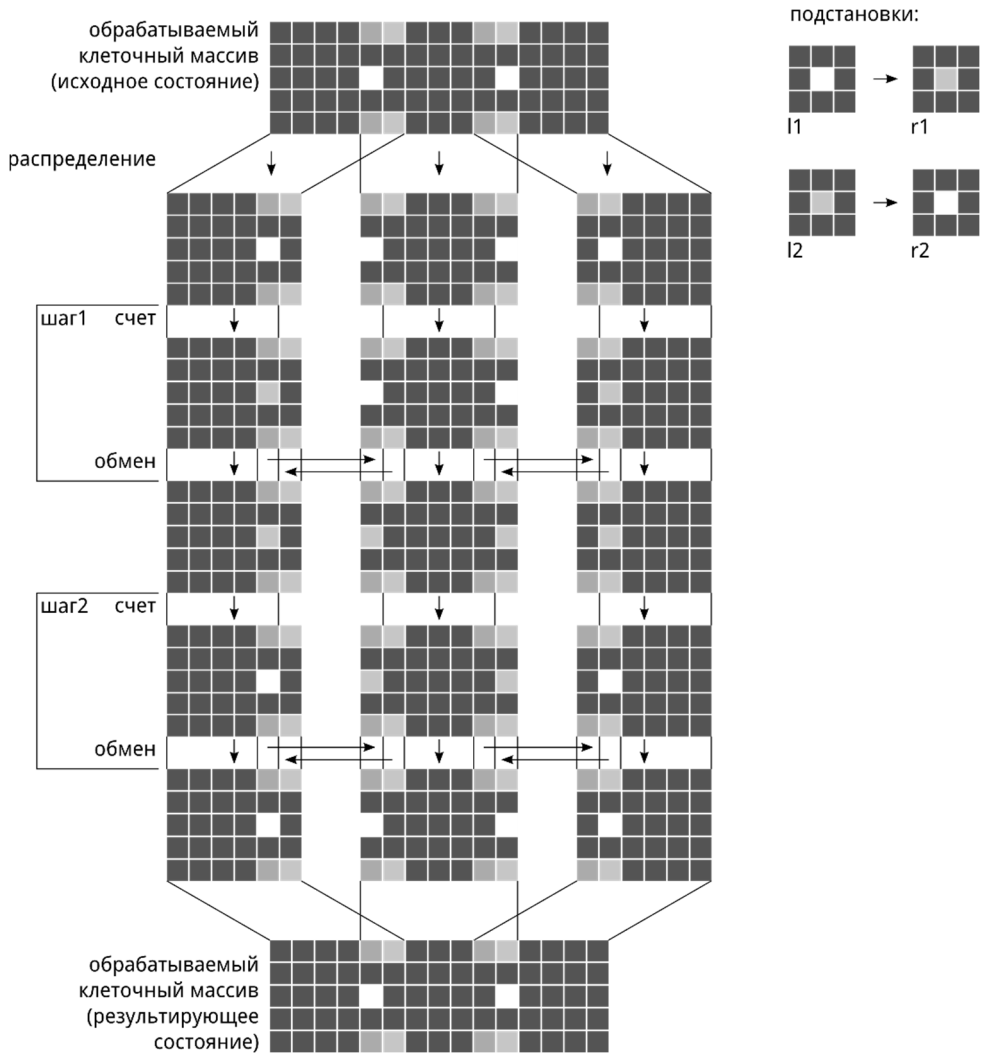

Рис. 4. Режим параллельного исполнения моделей с разрезанием объектов данных

При использовании режима разрезанием объектов данных перед параллельным исполнением происходит разделение обрабатываемого клеточного массива на части с перекрытием (с теневыми клетками). Их число определяется числом параллельных процессов обработки. Во многих случаях оно совпадает с числом счетных узлов. Будем рассматривать именно такой случай. Область наложения задает те клетки, которые размещаются в двух или более соседних счетных узлах. Геометрия этой области определяется топологией разрезания клеточного массива (линейка, кольцо, 2D, 3D решетка или тор) и максимальным размером окрестности имеющихся в модели правил подстановки, применяемых к данному клеточному массиву.

После разрезания клеточного массива, его части передаются соответствующим счетным узлам, и начинается параллельное исполнение. Оно происходит итеративно до тех

пор, пока есть клетки, значение которых менялось на последнем шаге. Каждый шаг состоит из двух фаз: счета и обмена. После завершения параллельного исполнения части клеточного массива собираются в целый массив в виде файла во внешней памяти.

В ParSIM/Omni вводятся дополнительные режимы распараллеливания, которые расширяют класс эффективно исполняемых МЗП моделей. Множество этих режимов может расширяться как разработчиком системы, так и ее пользователями имеющими навыки программирования, так как невозможно заранее предусмотреть все достаточные для моделей любого пользователя режимы. Изначально имеется реализация трех режимов параллельного исполнения. Первый из них заимствован из ParSIM/Win32 и рассмотрен выше. Два новых режима рассматриваются далее.

Режим с разбиением множества подстановок (рис. 5), являющийся примером функционального параллелизма, позволяет эффективно распараллеливать исполнение моделей, в которых есть большие окрестности в правилах подстановки, но при этом доля изменяемых клеток от общего их числа достаточно мала.

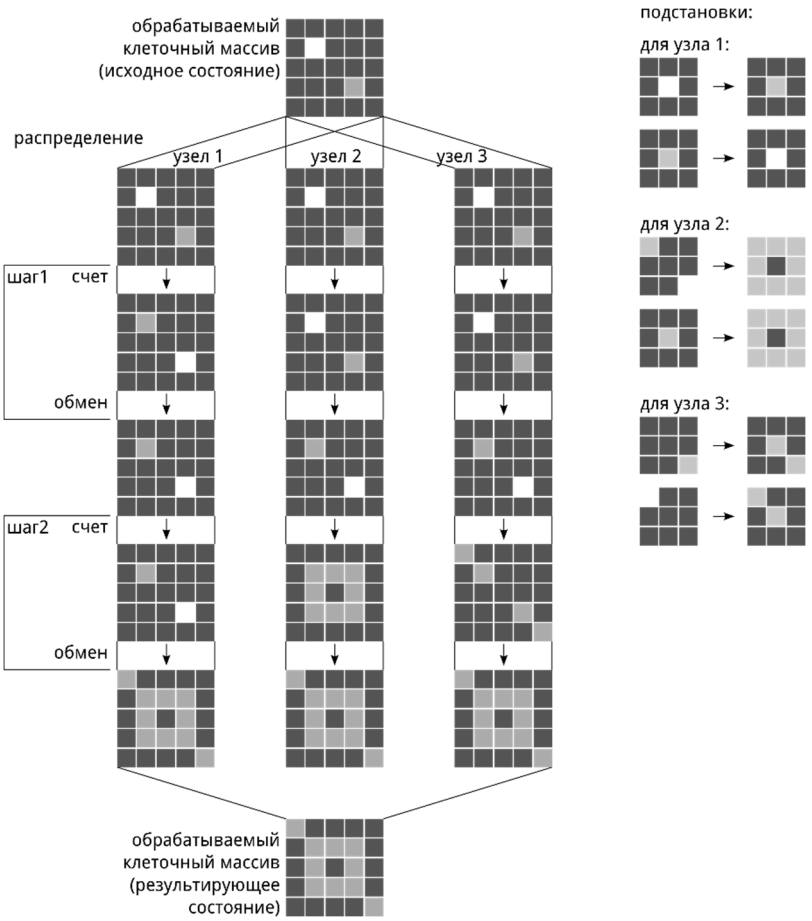

Рис. 5. Режим параллельного исполнения с разбиением множества подстановок

В этом режиме перед параллельным счетом все узлы вычислителя получают одинаковую полную копию обрабатываемого массива. Множество всех подстановок делится на непересекающиеся подмножества. Их число определяется числом параллельных процессов обработки. Во многих случаях оно совпадает с числом счетных узлов. Будем рассматривать именно такой случай. Каждый узел получает свой комплект подстановок. Параллельное исполнение происходит итеративно с синхронизацией вычислений между итерациями для проведения обменов. На фазе счета каждый узел отыскивает все применимые

подстановки из своего комплекта, вычисляет новые значения изменяемых клеток и формирует список всех измененных клеток. На фазе обмена каждый узел рассылает всем другим узлам значения всех модифицированных клеток. Если списки модифицированных клеток пустые на всех узлах, то результат получен, и параллельное вычисление прекращается.

Для случая, когда объем данных умеренный, но есть большой разброс времени обработки одной применимой подстановки, оправдано использование режима параллельного исполнения с разбиением обрабатываемых данных и вычислению по готовности данных (рис. 6).

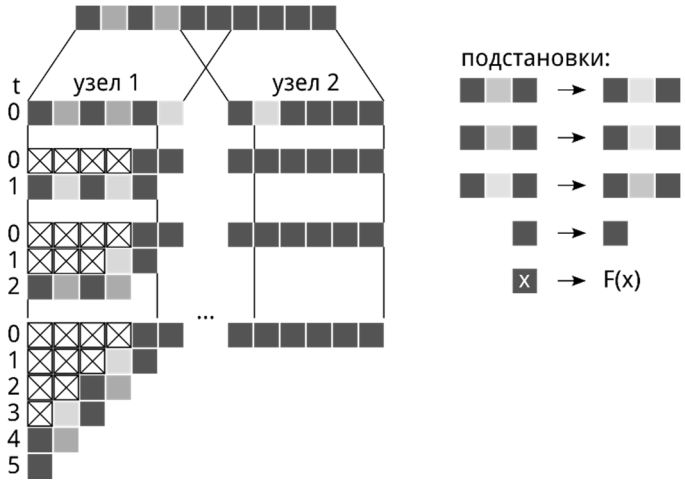

Рис. 6. Режим параллельного исполнения разбиением обрабатываемых данных и вычислению по готовности данных

В этом режиме возможно хранение значений одного элемента данных для разных итераций. Это дает возможность продолжать счет для внутренних областей клеток даже в тех случаях, когда новые значения клеток в области пересечения еще не получены от тех узлов, которые ответственны за обработку этих клеток. В этом режиме нет жесткого чередования фаз счета и обмена. Они могут реализовываться отдельными потоками команд, исполняемыми на счетном узле в режиме разделения времени.

В этом режиме возможно хранение значений одного элемента данных для разных итераций. Это дает возможность продолжать счет для внутренних областей клеток даже в тех случаях, когда новые значения клеток в области пересечения еще не получены от тех узлов, которые ответственны за обработку этих клеток. В этом режиме нет жесткого чередования фаз счета и обмена. Они могут реализовываться отдельными потоками команд, исполняемыми на счетном узле в режиме разделения времени.

Заключение

На настоящий момент реализована новая версия виртуальной машины и разработаны базовые системные алгоритмы для реализации остальных компонентов. Разрабатываются остальные компоненты верхнего уровня ParSIM/Omni. В первую очередь запланировано завершение реализации ParSIM/Omni на базе OpenCL.

Реализация ParSIM/Omni позволит расширить класс моделируемых в среде WinALT явлений и повысить точность результатов моделирования за счет увеличения размеров

обрабатываемых объектов данных. Приблизительная оценка предельных размеров клеточных массивов в ParSIM/Win32 составляет 30000×30000 клеток для 2D и 1000×1000×1000 для 3D. В ParSIM/Omni она составит 1000000×1000000 клеток для 2D и 10000×10000×10000 для 3D. Для многих моделей, например из области теории протекания [6], подобное увеличение размеров приведет к существенно более точным и качественным результатам моделирования. Для ряда других моделей, таких как каталитические реакции получение результата в принципе невозможно при малом числе клеток. В [7] указывается, что требуемое число взаимодействующих молекул составляет 10^{20}.

Также, реализация ParSIM/Omni позволит увеличить объем производимых вычислений как за счет возможности использования высокопроизводительных параллельных вычислительных комплексов, так и за счет возможности виртуальной машины сохранять состояние процесса моделирования и возобновлять моделирование после сбоев различного характера.

Литература

1. Тоффоли, Т. Машины клеточных автоматов / Т. Тоффоли, Н. Марголус – М.: Мир, 1991 – 278 с.
2. Inman, R. Cellular Automata FAQ - Cellular Automata Software / R. Inman, H.H. Chou, et al. // URL: http://cafaq.cafaq.com/soft (дата обращения: 06.01.2013).
3. Остапкевич, М.Б. Сайт среды моделирования WinALT. / М.Б. Остапкевич, С.В. Пискунов. URL: http://winalt.sscc.ru (дата обращения: 06.01.2013).
4. Ostapkevich, M. The Construction of Simulation Models of Algorithms and Structures with Fine-Grain Parallelism in WinALT / M. Ostapkevich, S. Piskunov // PaCT 2011, LNCS – Springer-Verlag, 2011. – Vol. 6873. – P. 192–203.
5. Naor, M. Visual Cryptography / M. Naor, A. Shamir // Advances in Cryptology, EUROCRYPT'94. – 1995. – P. 1–12.
6. Эфрос, А.Л. Физика и геометрия беспорядка / А.Л. Эфрос // Библиотека Квант. - вып. 19. – М: Наука, 1982. – 268 с.
7. Latkin, E.L. Manifestation of the adsorbed co-diffusion anisotropy caused by the structure properties of the Pd(110) - (1x2) surface on the oscillatory behavior during co-oxidation reaction - Monte-Carlo model // Chemistry for Sustainable Development. – 2003. – No. 11 – P. 173–180.

Остапкевич Михаил Борисович, м.н.с., ИВМиМГ СО РАН, ostap@ssd.sscc.ru.

PARALLEL IMPLEMENTATION OF FINE-GRAIN ALGORITHMS IN WINALT SYSTEM

M.B. Ostapkevich Institute of Computational Mathematics and Mathematical Geophysics (Novosibirsk, Russian Federation),

A brief description of a simulation system for algorithms and structures with fine-grain parallelism WinALT is given. The distinguishing features of the system are a visual construction and debugging of models as well as its orientation to a wide range of algorithms and structures with fine-grain parallelism rather than only to a cellular automaton along with its certain extensions. The existing subsystem of parallel execution, which is capable to execute models using a cluster of Windows hosts, is considered. The requirements to a new implementation of the subsystem for parallel execution are formulated. This new implementation is oriented to a wide range of modern parallel computers including hybrid clusters. The architecture of the subsystem is proposed. Its modes of parallel execution are considered. The plans of further development of the system are outlined.

Keywords: fine-grain parallelism, simulation, parallel programming, simulation system, parallel substitutions algorithm.

References

1. Toffoli T., Margolus N. Mashini kletochnikh avtomatov [Cellular automata machines]. Moscow: Mir, 1991. 278 p.
2. Inman R., Chou H.H. et al. Cellular Automata FAQ - Cellular Automata Software. URL: http://cafaq.cafaq.com/soft (accessed: 06.01.2013).
3. Ostapkevich M., Piskunov S. WinALT simulation environment site. URL: http://winalt.sscc.ru (accessed: 06.01.2013).
4. Ostapkevich M., Piskunov S. The Construction of Simulation Models of Algorithms and Structures with Fine-Grain Parallelism in WinALT // PaCT 2011, LNCS. Springer-Verlag, 2011. Vol. 6873. P. 192–203.
5. Naor M., Shamir A. Visual Cryptography // Advances in Cryptology, EUROCRYPT'94. 1995. P. 1–12.
6. Efros A.L. Fizika i geometrija besporjadka [Physics and geometry of chaos] // Kvant library. Vol. 19. Moscow: Nauka, 1982. 268 p.
7. Latkin E.L., Elokhin V.I., Matveev A.V., Gorodetskii V.V. Manifestation of the adsorbed co-diffusion anisotropy caused by the structure properties of the Pd(110) – (1x2) surface on the oscillatory behavior during co-oxidation reaction - Monte-Carlo model // Chemistry for Sustainable Development. 2003. No 11. P. 173–180.

Поступила в редакцию 7 января 2013 г.

УДК 517.948

ОБ ОЦЕНКЕ ПОГРЕШНОСТИ В ТОЧКЕ ПРИ РЕШЕНИИ ОБРАТНЫХ ЗАДАЧ

В.П. Танана, Т.С. Камалтдинова

Предложен новый подход к оценке погрешности нелинейного метода проекционной регуляризации в точке при решении некорректных задач. Приведено сравнение этой погрешности на множестве.

Ключевые слова: некорректная задача, метод проекционной регуляризации, оценка погрешности, модуль непрерывности, гильбертово пространство.

Введение

При оценке погрешности методов решения некорректно поставленных задач приходится сталкиваться с трудностью, связанной с неопределенностью точного решения. Выход из этого положения заключается в том, что дополнительно вводится класс корректности и предполагается, что точное решение задачи принадлежит этому классу. Затем (см. [1]) оценка погрешности определяется на этом классе. Естественно, что при таком подходе, для получения оценки, из класса корректности выбирается «худший» элемент. Этот подход приводит к тому, что реальная погрешность оказывается меньше теоретической.

В настоящей работе, при условии кусочной гладкости точного решения, сделана попытка сравнить оценку погрешности в точке с оценкой на классе. Для этой цели использован нелинейный метод проекционной регуляризации [2].

1. Постановка задачи

Пусть $L_2(-\infty,\infty)$ - комплексное пространство, а $Z \subset L_2(-\infty,\infty)$. Элемент $u(x) \in Z$ тогда и только тогда, когда

$$u(x) = \sum_{i=1}^{n} \varphi_i(x) + \psi(x),$$

где

$$\varphi_i(x) = \begin{cases} a_i\left(x^2 - x_i^2\right), & |x| \le x_i \\ 0, & |x| > x_i \end{cases},$$

a_i u $x_i > 0$, a $\psi(-x) = \psi(x)$ u

$$\psi(x) \in W_2^{3/2}(-\infty,\infty).$$

Рассмотрим линейные, неограниченные, замкнутые операторы T u G с областями определения $D(T)$ u $D(G) \subset L_2(-\infty,\infty)$ и множествами значений $R(T)$ u $R(G) \subset L_2(-\infty,\infty)$ такие, что

$$\overline{R(T) \bigcap D(G)} = L_2(-\infty,\infty),$$

где $\overline{R(T) \bigcap D(G)}$ - замыкание множества $R(T) \bigcap D(G)$.

Поставим задачу вычисления значения Tf_0 оператора T в точке $f_0 \in D(T)$

$$Tf = u. \qquad (1)$$

Предположим, что при $f = f_0$ элемент $u_0 = Tf_0$ принадлежит множеству Z, но точное значение f_0 нам не известно, а вместо него даны $f_\delta \in L_2(-\infty,\infty)$ и $\delta > 0$ такие, что

$$\|f_\delta - f_0\| \le \delta. \tag{2}$$

Требуется, используя априорную информацию f_δ и δ, определить приближенное значение $u_\delta \in L_2(-\infty,\infty)$ задачи (1), (2) и оценить уклонение $\|u_\delta - u_0\|$.

2. Основные понятия

Определение 1. Семейство $\{T_\delta : 0 < \delta \le \delta_0\}$ операторов T_δ, непрерывно отображающих $L_2(-\infty,\infty)$ в себя, будем называть методом решения задачи (5), (6), если для любого $u_0 \in Z$

$$\sup_{f_\delta}\left\{\|T_\delta f_\delta - u_0\| : f_\delta \in L_2(-\infty,\infty), \|f_\delta - T^{-1}u_0\| \le \delta\right\} \to 0 \text{ при } \delta \to 0.$$

Введем оценку погрешности $\Delta_\delta(u_0,\delta)$ метода $\{T_\delta : 0 < \delta \le \delta_0\}$ в точке $u_0 \in Z$

$$\Delta_\delta(u_0,\delta) = \sup_{f_\delta}\left\{\|T_\delta f_\delta - u_0\| : f_\delta \in L_2(-\infty,\infty), \|f_\delta - T^{-1}u_0\| \le \delta\right\}.$$

Обозначим через M_r множество, определяемое формулой

$$M_r = \{u : u \in R(T) \cap D(G), \|Gu\| \le r\}.$$

На нем определим модуль непрерывности $\omega(\tau,r)$ (см. [1]), формулой

$$\omega(\tau,r) = \sup_{f}\{\|Tf\| : f \in T^{-1}(M_r), \|f\| \le \tau\}; \quad \tau, r > 0, \tag{3}$$

а также оценку погрешности $\Delta_\delta(M_r,\delta)$ метода $\{T_\delta : 0 < \delta \le \delta_0\}$ на множестве M_r

$$\Delta_\delta(M_r,\delta) = \sup_{u_0, f_\delta}\left\{\|T_\delta f_\delta - u_0\| : u_0 \in M_r,\ f_\delta \in L_2(-\infty,\infty), \left\|f_\delta - T^{-1}u_0\right\| \le \delta\right\}.$$

Известно, что для любого метода $\{T_\delta : 0 < \delta \le \delta_0\}$ справедлива оценка

$$\Delta_\delta(M_r,\delta) \ge \omega(\delta,r); \quad 0 < \delta \le \delta_0. \tag{4}$$

3. Нелинейный метод проекционной регуляризации

Для решения задачи (1), (2) перейдем от функций $u(x)$ и $f(x)$ к Фурье-образам $\hat{u}(\xi) = F[u(x)]$ и $\hat{f}(\xi) = F[f(x)]$, где F – преобразование Фурье в $L_2(-\infty,\infty)$.

При этом переходе операторы T и G перейдут в операторы $\hat{T}$ и $\hat{G}$, действующие в пространстве $L_2(-\infty,\infty)$ и определяемые действительными функциями $t(\xi)$ и $g(\xi)$.

$$\hat{T}\hat{f}(\xi) = t(\xi)\hat{f}(\xi); \hat{f}(\xi) \in D(\hat{T})$$

$$D(\hat{T}) = \left\{\hat{f}(\xi) : \hat{f}(\xi) \in L_2(-\infty,\infty), t(\xi)\hat{f}(\xi) \in L_2(-\infty,\infty)\right\}$$

и

$$\hat{G}\hat{u}(\xi) = g(\xi)\hat{u}(\xi); \hat{u}(\xi) \in D(\hat{G}),$$

$$D(\hat{G}) = \left\{\hat{u}(\xi) : \hat{u}(\xi) \in L_2(-\infty,\infty), g(\xi)\hat{u}(\xi) \in L_2(-\infty,\infty)\right\}$$

При этом множество M_r перейдет в $\hat{M}_r = F[M_r]$, определяемое

$$\hat{M}_r = \left\{\hat{u}(\xi) : \hat{u}(\xi) \in R(\hat{T}) \cap D(\hat{G}), \left\|\hat{G}\hat{u}(\xi)\right\| \le r\right\}.$$

На $\hat{M}_r$, аналогично (3), определим модуль непрерывности $\hat{\omega}(\tau,r)$ формулой

$$\hat{\omega}(\tau,r)=\sup_{\hat{f}}\left\{\left\|\hat{T}\hat{f}\right\|:\hat{f}\in\hat{T}^{-1}\left(\hat{M}_r\right),\left\|\hat{f}\right\|\le\tau\right\}.$$

Из теоремы Планшереля следует, что задача (1), (2) перейдет в метрически эквивалентную задачу

$$\hat{T}\hat{f}=\hat{u}. \tag{5}$$

Предположим, что при $\hat{f}=\hat{f}_0$ элемент $\hat{u}_0=\hat{T}\hat{f}_0$ принадлежит множеству $\hat{Z}=F[Z]$, но вместо $\hat{f}_0$ нам известны $\hat{f}_\delta\in L_2(-\infty,\infty)$ и $\delta>0$ такие, что

$$\left\|\hat{f}_\delta-\hat{f}_0\right\|\le\delta. \tag{6}$$

В дальнейшем предположим, что функции $t(\xi)$ и $g(\xi)$ непрерывные, четные, положительные и строго возрастающие на $[0,\infty)$. Кроме того, $t(\xi)\to\infty$ и $g(\xi)\to\infty$ при $\xi\to\infty$.

Нелинейный метод проекционной регуляризации [2] зададим семейством $\left\{\hat{T}_\delta:0<\delta\le\delta_0\right\}$ операторов $\hat{T}_\delta$, непрерывно отображающих $L_2(-\infty,\infty)$ в себя и определяемых формулами

$$\hat{T}_\delta\hat{f}_\delta(\xi)=\begin{cases}\hat{T}_{\hat{\alpha}\left(\hat{f}_\delta,\delta\right)}\hat{f}_\delta(\xi); & \left\|\hat{f}_\delta(\xi)\right\|>3\delta\\ 0; & \left\|\hat{f}_\delta(\xi)\right\|\le3\delta\end{cases},$$

где $\hat{T}_\alpha\hat{f}(\xi)=\begin{cases}\hat{T}\hat{f}(\xi); & \|\xi\|\le\alpha\\ 0; & \|\xi\|>\alpha\end{cases}$, $\alpha>0$, а $\hat{\alpha}\left(\hat{f}_\delta,\delta\right)$ определяется уравнением

$$\int_\alpha^\infty\left|\hat{f}_\delta(\xi)\right|^2d\xi=9\delta^2.$$

Таким образом, приближенное решение $\hat{u}_\delta(\xi)$ задачи (5), (6) определим формулой

$$\hat{u}_\delta(\xi)=\hat{T}_\delta\hat{f}_\delta(\xi).$$

4. Оценка погрешности $\left\|\hat{u}_\delta-\hat{u}_0\right\|$

Лемма 1. Если $\hat{Z}\subset\bigcup_{n=1}^{\infty}n\hat{M}_r$, а функция $\frac{g^2(\xi)}{1+\xi^4}$ убывает на $[0,\infty)$, то

$$\int_0^\infty\frac{g^2(\xi)}{1+\xi^4}d\xi<\infty.$$

Теорема 1. Предположим, что выполнены условия леммы 1, $\hat{u}_0\in\hat{Z}\bigcap\hat{M}_r$ и для любого $k>0$ найдутся числа $d_k,d_k'\,u\,\xi_k>0$ такие, что для любого ξ, удовлетворяющего условию $|\xi|>\xi_k$ справедливы соотношения

$$d_k\le\frac{g(\xi)}{g(k\xi)}\le d_k'.$$

Тогда $\frac{\Delta_\delta(\hat{u}_0,\delta)}{\hat{\omega}(\delta,r)}\to0$ при $\delta\to0$.

Теорема 2. Пусть выполнены все условия теоремы 1. Тогда

$\frac{\Delta_\delta(\hat{u}_0,\delta)}{\Delta_\delta\left(\hat{M}_r,\delta\right)}\to0$ *при* $\delta\to0$.

5. Решение обратной задачи Коши для уравнения теплопроводности

Рассмотрим уравнение

$$\frac{\partial u(x,t)}{\partial t} = \frac{\partial^2 u(x,t)}{\partial x^2}; -\infty < x < \infty, t \in (0,t_0], t_0 > 0, \tag{7}$$

где $u(x,t) \in C\{(-\infty,\infty)\times[0,t_0]\} \cap C^{2,1}\{(-\infty,\infty)\times(0,t_0]\}$ для любого $t \in (0,t_0]$, $u(x,t), u'_x(x,t), u''_{xx}(x,t) \in L_2(-\infty,\infty) \cap L_1(-\infty,\infty)$ и существует $\chi(x) \in L_1(-\infty,\infty)$ такая, что почти для любого $t \in (0,t_0]$

$$|u(x,t)| + \left|\frac{\partial u(x,t)}{\partial t}\right| \le \chi(x), \quad x \in (-\infty,\infty).$$

Пусть нам дано распределение $\varphi(x) \in L_2(-\infty,\infty) \cap L_1(-\infty,\infty)$ в момент времени t_0

$$u(x,t_0) = \varphi(x); \quad -\infty < x < \infty, \tag{8}$$

а начальное распределение $u_0(x)$

$$u(x,0) = u_0(x); \quad -\infty < x < \infty \tag{9}$$

требуется определить.

Предположим, что при $\varphi(x) = \varphi_0$ существует $u_0(x) \in Z$, такое, что решение задачи (7), (9) удовлетворяет условию $u(x,t_0) = \varphi_0(x)$, но точное значение $\varphi_0(x)$ нам не известно, а вместо него даны некоторое приближение $\varphi_\delta(x) \in L_2(-\infty,\infty) \cap L_1(-\infty,\infty)$ и $\delta > 0$ такие, что

$$\|\varphi_\delta(x) - \varphi_0(x)\|_{L_2} \le \delta. \tag{10}$$

Требуется, используя исходные данные (φ_δ, δ) задачи (7), (8), (10), определить приближенное решение $u_\delta(x) \in L_2(-\infty,\infty)$ и оценить величину $\|u_\delta(x) - u_0(x)\|_{L_2}$.

Применив к задаче (7), (8), (10) преобразование Фурье F, сведем ее к задаче вычисления значений неограниченного оператора $\hat{T}$

$$\hat{T}\hat{\varphi}_\delta(\lambda) = e^{\lambda^2 t_0}\hat{\varphi}_\delta(\lambda) = \hat{u}(\lambda). \tag{11}$$

$$\|\hat{\varphi}_\delta(x) - \hat{\varphi}_0(x)\|_{L_2} \le \delta. \tag{12}$$

Приближенное решение $\hat{u}_\delta(\lambda)$ задачи (11), (12) определим формулой

$$\hat{u}_\delta(\lambda) = \begin{cases} e^{\lambda^2 t_0}\hat{\varphi}_\delta(\lambda); & |\lambda| \le \alpha(\hat{\varphi}_\delta,\delta) \\ 0; & |\lambda| > \alpha(\hat{\varphi}_\delta,\delta) \end{cases},$$

где $\alpha(\hat{\varphi}_\delta,\delta)$ удовлетворяет условию

$$\int\limits_{\alpha(\hat{\varphi}_\delta,\delta)}^{\infty} |\hat{\varphi}_\delta(\lambda)|^2 d\lambda = 9\delta^2$$

Пусть $\hat{Z} = F[Z]$, $\hat{M}_r$ определим формулой

$$\hat{M}_r = \left\{\hat{u}(\lambda) : \hat{u}(\lambda) \in R(\hat{T}), \|g(\lambda)\hat{u}(\lambda)\| \le r\right\},$$

где $g(\lambda) = g(-\lambda), g \in C(-\infty,\infty), g(\lambda) > 0,$ строго возрастая, стремится к бесконечности.

$$\hat{\omega}(\delta,r)=\sup_{\hat{\varphi}}\left\{\left\|\hat{T}\hat{\varphi}\right\|:\hat{\varphi}\in\hat{T}^{-1}\left(\hat{M}_r\right),\|\hat{\varphi}\|\le\delta\right\},$$

$$\Delta_\delta\left(\hat{u}_0,\delta\right)=\sup_{\hat{\varphi}_\delta}\left\{\left\|\hat{u}_\delta\left(\lambda\right)-\hat{u}_0\left(\lambda\right)\right\|:\hat{\varphi}_\delta\in L_2\left(-\infty,\infty\right),\left\|\hat{\varphi}_\delta-\hat{T}^{-1}\hat{u}_0\right\|\le\delta\right\},$$

а $\Delta_\delta\left(\hat{M}_r,\delta\right)=\sup\limits_{\hat{u}_0,\hat{\varphi}_\delta}\left\{\left\|\hat{u}_\delta-\hat{u}_0\right\|:\hat{u}_0\in M_r,\hat{\varphi}_\delta\in L_2\left(-\infty,\infty\right),\left\|\hat{\varphi}_\delta-\hat{T}^{-1}\hat{u}_0\right\|\le\delta\right\}$.

Теорема 3. Предположим, что $\hat{Z}\subset\bigcup\limits_{n=1}^{\infty}n\hat{M}_r$, $\dfrac{g^2\left(\lambda\right)}{1+\lambda^4}$ убывает на $\left[0,\infty\right)$, $\hat{u}_0\left(\lambda\right)\in\hat{Z}\cap\hat{M}_r$ и $\left\|\hat{\varphi}_\delta\right\|>3\delta$.

Тогда

$$\frac{\Delta_\delta\left(\hat{u}_0,\delta\right)}{\hat{\omega}\left(\delta,r\right)}\to 0\quad при\quad \delta\to 0.$$

Теорема 4. При выполнении условий теоремы 3 справедливо соотношение

$$\frac{\Delta_\delta\left(\hat{u}_0,\delta\right)}{\Delta_\delta\left(\hat{M}_r,\delta\right)}\to 0\quad при\quad \delta\to 0.$$

Заключение

Введена оценка погрешности в точке при решении некорректных задач. Приведено сравнение этой погрешности с погрешностью на множестве. Другие методы решения задачи (7)–(9) и ее нелинейного варианта рассмотрены, например, в работе [3].

Литература

1. Ivanov, V.K. Theory of Linear Ill-Posed Problems and its Applications / V.V. Vasin, V.P. Tanana. – VSP, 2002
2. Танана, В.П. Об оценке погрешности приближенного решения одной обратной задачи в классе кусочно-гладких функций / А.Б. Бредихина, Т.С. Камалтдинова // Труды Института математики и механики УрО РАН. – 2012. – Т. 18, №1. – С. 281–288.
3. Табаринцева Е.В. Об оценке погрешности метода квазиобращения при решении задачи Коши для полулинейного дифференциального уравнения / Е.В. Табаринцева // Сиб. журнал вычисл. матем. – 2005. – Т. 8, № 3. – С. 259–271.

Танана Виталий Павлович, заведующий кафедрой вычислительной математики, Южно-Уральский государственный университет, tvpa@susu.ac.ru

Камалтдинова Татьяна Сергеевна, старший преподаватель, Южно-Уральский государственный университет, KamaltdinovaTS@ mail.ru.

ON POINT-WISE ERROR ESTIMATE IN SOLVING INVERSE PROBLEMS

V.P. Tanana, South Ural State University (Chelyabinsk, Russian Federation),
T.S. Kamaltdinova, South Ural State University (Chelyabinsk, Russian Federation)

A new approach to the point-wise error estimation for the projection ill posed problems is suggested in the article. We compare point-wise error estimate with the error estimate on a set.

Keywords: ill-posed problem, method of projective regularization, error estimation, module of coninuity.

References

1. Ivanov V.K., Vasin V.V., Tanana V.P. Theory of Linear Ill-Posed Problems and its Applications. VSP, 2002.
2. Tanana V.P., Bredikhina A.B., Kamaltdinova T.S. On an error estimate for an approximate solution for an inverse problem in the class of piecewise smooth functions. Supplement to Proceedings of the Steklov Institute of Mathematics and Mechanics Ural Branch of RAS. 2012. Vol. 18, No. 1. P. 281–288.
3. Tabarintseva E.V. On error estimate of the quasi-reversibility method to solve the Cauchy problem for a semielinear differential equation. Siberian journal of numerical mathematics. 2005. Vol. № 3. P. 259–271.

Поступила в редакцию 14 января 2013 г.

УДК 004.046

ПРИМЕНЕНИЕ КОНЦЕПЦИИ АКТИВНЫХ ХРАНИЛИЩ В ЗАДАЧАХ ОБРАБОТКИ ДАННЫХ СЕЙСМИЧЕСКИХ НАБЛЮДЕНИЙ[1]

Е.О. Тютляева, А.А. Московский, С.С. Конюхов, Е.А. Курин

Предложен подход для организации распределенной обработки сейсмических данных на базе свободно распространяемого пакета Seismic Un*x и системы активного хранения данных с использованием TSim и ФС Lustre. В работе рассмотрены ключевые проблемы обработки сейсмических данных, и для каждой предложено и обосновано использование соответствующего инструмента из арсенала системы активного хранения данных. Обработка данных непосредственно на узлах хранения позволяет продемонстрировать значительную эффективность за счет минимизации количества дорогостоящих операций передачи данных по сети. Проведено исследование производительности разработанного программного прототипа по обработке сейсмических данных в системе активного хранения для оценки перспектив полноценной интеграции.

Ключевые слова: активные хранилища, обработка сейсмических данных, распределенное хранение и обработка данных.

Введение

Решение задач обработки данных, полученных при сейсмической разведке, предъявляет высокие требования к вычислительным ресурсам. С одной стороны это связано с большими объемами данных сейсмических наблюдений и их первичной обработки, которые достигают сотен терабайт. С другой стороны сложность обусловлена свойствами применяемых на практике математических методов, требующих произвести большой объем вычислений в достаточно сжатые сроки.

Например, в 2009 г. на месторождении Карачаганак (Казахстан) проводились трехмерные наблюдения на площади около 900 км^2. При этом были зарегистрированы около трех миллиардов записей (трасс). Общий объем данных превысил 100 Тб [1]. Другим примером может служить работа [2], где приводится сопоставление различных методов построения глубинного изображения среды по сейсмическим данным и необходимых вычислительных ресурсов.

Приведенные в работах [1, 2] примеры показывают, что в ряде случаев применение суперкомпьютеров может значительно улучшить качество результатов обработки сейсмических данных, а также обеспечить решение задач в экономически оправданные сроки.

Многообразие решаемых задач в сейсморазведке порождает многообразие применяемых вычислительных методов и программ обработки сейсмических данных. Одним из популярных пакетов по обработке сейсмических данных является свободно распространяемый пакет Seismic Un*x [3], создание которого началось более 24 лет назад в Center for Wave Phenomena Colorado School of Mines. В настоящее время данный пакет содержит большинство необходимых операций над сейсмическими данными и активно используется и обновляется специалистами по сейсмологии со всего мира. Seismic Un*x позволяет строить комплексные задания обработки данных из набора простых процедур при помощи конвейеров Unix.

[1]Статья рекомендована к публикации программным комитетом международной научной конференции «Научный сервис в сети Интернет: Поиск новых решений — 2012»

При обработке сейсмических данных при помощи стандартного набора процедур из пакета Seismic Un*x возможна обработка отдельных порций данных независимо друг от друга. Для этого класса задач узким местом является подсистема ввода-вывода.

Одним из возможных подходов к преодолению данного узкого места является концепция систем активного хранения данных. Основная идея системы активного хранения данных в контексте параллельных файловых систем заключается в использовании вычислительных ресурсов узлов хранения для обработки данных, расположенных на данном узле. Предложенный подход позволяет использовать вычислительный кластер как для хранения, так и обработки данных.

Обработка данных непосредственно на узлах хранения позволяет продемонстрировать значительную эффективность за счет минимизации количества дорогостоящих операций передачи данных по сети. Дополнительным достоинством при обработке значительных объемов данных может являться своевременная доставка данных для обработки к вычислительному процессору, и связанное с этим снижение вероятности простоя вычислительных ресурсов в ожидании доставки данных.

Описанная концепция реализована в системе активного хранения данных, базирующейся на библиотеке шаблонных классов C++ TSim для распараллеливания вычислений и кластерной файловой системе Lustre для обеспечения высокопроизводительного масштабируемого ввода-вывода. Разработанный прототип системы и опыт ее использования для хранения и обработки данных дистанционного зондирования уже были представлена на конференции «Научный сервис в сети интернет» [4]. В развитие предшествующих результатов, в данной статье будет представлен ряд модификаций данной системы и интеграция данного подхода со свободно распространяемым пакетом Seismic Un*x, предназначенным для обработки сейсмических данных.

1. Структура данных и вычислительных модулей в Seismic Un*x

Базовой структурной единицей хранения сейсмических данных в формате Seismic Un*x (SU) является трасса. Каждая трасса содержит заголовок фиксированного размера и бинарные данные, отражающие результаты измерения с описанными в заголовке параметрами. Файл SU содержит набор трасс с результатами различных измерений, проведенных в ходе эксперимента. Большинство программ из пакета Seismic Un*x обрабатывают трассы (все, или отвечающие определенным критериям) последовательно, независимо друг от друга. Типовая схема работы с сейсмическими данными приведена на рис. 1.

Поскольку, как видно из типовой схемы, трассы в большинстве случаев обрабатываются независимо, возможно эффективно организовать распределенные вычисления в системе активного хранения данных.

```
/* Считывание первой трассы; инициализация основных параметров */
if (!gettr(&tr)) err("can't read first trace");
if (!tr.dt) err("dt header field must be set");
/* Loop over traces */
do {
 /* Обработка трассы, подготовка модифицированного результата на запись*/
}
puttr(&tr);  //запись обработанной трассы в файл результата
} while (gettr(&tr)); //считывание следующей трассы для обработки
```

Рис. 1. Типовая схема работы с сейсмическими данными

2. Принципы интеграции пакета Seismic Un*x и системы активного хранения

Большое значение при интеграции имела возможность организации распределенных вычислений без модификации исходных кодов пакета Seismic Un*x. Система активного хранения на базе TSim предоставляет такую возможность при обработке разных файлов, распределенных по узлам вычислительного кластера.

Поскольку данные сейсмических наблюдений обычно представлены в виде одного файла в формате SU, для удобства распределенной обработки в системе активного хранения файл был разбит на фрагменты, количество которых было выбрано кратным количеству узлов вычислительного кластера. Разбиение выполняется при помощи утилиты suwind пакета Seismic Un*x.

Полученные фрагменты исходного файла были размещены в параллельной файловой системе Lustre для последующего хранения и обработки. Перед размещением файлов была задана схема разбиения таким образом, чтобы каждый полученный фрагмент лежал целиком на одном узле, при этом чтобы все фрагменты были распределены равномерно по всем узлам вычислительного кластера.

Предложенный подход имеет ряд преимуществ:

- допускается постепенный перенос данных и обработки в кластер: часть данных и обработчиков могут находиться в «традиционном» серверном Seismic Un*x, а часть быть реализована в активном хранилище;
- размещение данных прозрачно для ОС и для пользователя: для доступа к файлам используются средства ФС Lustre;
- за счет возможностей ФС Lustre возможна интеграция в высокопроизводительные вычислительные комплексы (например, за счет интерфейса MPI-IO).

Для дальнейшей обработки файлов достаточно вызывать планировщик системы активного хранения на базе библиотеки TSim. При вызове планировщика ему необходимо передать полный путь к обрабатываемым файлам данных, расположенным в ФС Lustre, конвейер обработки и список файлов, к которым необходимо применить данный конвейер. Далее применение заданного конвейера обработки к заданным файлам будет выполнено автоматически, в соответствии с логикой активного хранения данных. Упрощенная схема распределения данных по узлам вычислительного кластера и последующей обработки изображена на рис. 2.

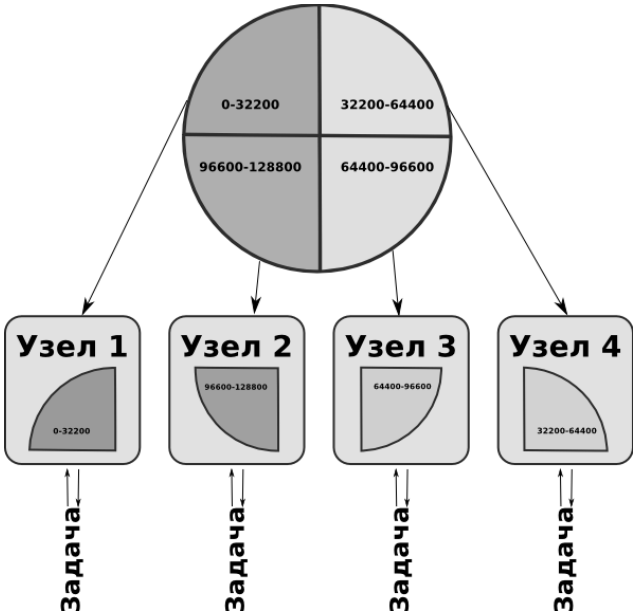

Рис. 2. Схема распределения данных по узлам

Алгоритм распределенной обработки состоит из следующих этапов:

- На первом шаге для обработки каждого файла порождается отдельная параллельная задача.
- Для каждой задачи планировщик системы активного хранения определяет схему расположения обрабатываемого файла в ФС Lustre, в частности, на каком устройстве хранения расположен данный файл.
- По конфигурационным настройкам Lustre планировщик определяет, к какому вычислительному узлу относится определенное устройство хранения и перенаправляет вычисления на заданный узел.

3. Предварительная оценка производительности

Описанная схема распределенной обработки была применена для обработки файла сейсмических данных размером в 767 Мб на вычислительном кластере, характеристики которого приведены в таблице.

Табл. Конфигурация вычислительного кластера в ИПС им. А.К. Айламазяна РАН

Количество узлов	7
Тип процессоров	Intel Xeon E5410 (12M L2 Cache, 2,33 ГГц, 1333 МГц FSB, 4 ядра) — 10 шт., на 5 узлах, Intel Xeon X5570 (8M Cache, 2,93 ГГц, 6,40 ГТ/с Intel QPI, 4 ядра) — 4 шт., на 2 узлах
Количество ядер	56
Объем ФС Lustre	6,3 Тб
Интерконнект	Gigabit Ethernet
Форм-фактор	Full ATX

Файл данных был разбит на 7 частей по 110 Мб в соответствии с доступным количеством вычислительных узлов кластера. Полученные фрагменты были распределены по одному на каждый узел средствами ФС Lustre.

Конвейер обработки содержал только те команды, которые можно применять к каждой трассе независимо и имел вид представленный на рис. 3.

```
sufilter f=3,7,23,30 amps=0.5,1,1,0 < filei.su|
supef maxlag=MAXLAG_SPIKING |
sunmo cdp=600,1200 tnmo=0,1.2,1.8,2.4,2.9,3.6,4.0,4.3 vnmo=1500,150
0,1630,1770,1925,2080,2350,2280 tnmo=0,0.7,1.4,2.06,2.46,3.37,3.88,4.07
vnmo=1500,1500,1670,1780,1900,2130,2250,2220 |
sugain tpow=2 agc=1 | sustack |
sufilter f=3,7,23,30 amps=0.5,1,1,0
```

Рис. 3. Использованный конвейер обработки

Данный конвейер применялся параллельно ко всем частям исходного файла данных.

Последовательная обработка целого файла при помощи приведенного конвейера занимает 21,784 секунд.

Параллельная обработка по указанной схеме в системе активного хранения занимает 4,400 секунд.

Параллельная обработка при помощи планировщика «барабанного» типа (не учитывающего расположения файлов) занимает 11.009 секунд. Важно отметить, что файловая система Lustre позволяет клиенту выполнять операции с данными вне зависимости от того, где они расположены, что позволило обеспечить корректную обработку данных в случае «барабанного» планировщика без дополнительных операций с данными.

Приведенные значения измерения времени являются усредненными результатами по 10 экспериментам.

Следует отметить, что даже при таком незначительном времени обработки файла удалось получить повышение эффективности при использовании системы активного хранения данных, несмотря на необходимые накладные расходы, связанные с планированием и пересылкой задач на вычислительные узлы. Мы надеемся, что при проведении экспериментов с большими объемами данных, удастся получить ускорение, близкое к линейному.

Полученные предварительные результаты продемонстрировали целесообразность использования системы активного хранения для обработки сейсмических данных.

4. Сортировка данных в активном хранилище

Программы из набора Seismic Un*x, которые обладают более сложными зависимостями по данным чем описано в предыдущем разделе, очевидно, более сложны для распараллеливания. Характерным примером является операция сортировки трасс в файле (модуль susort), которая используется в большинстве конвейеров обработки. При сортировке файла сейсмических данных при помощи операции susort сначала считываются значения тех ключей, по которым будет проводиться сортировка. Затем множество ключей сортируется в соответствии с заданным порядком, при этом с каждым ключом связан номер трассы,

к которой данный ключ относится. Наконец, трассы исходного файла считываются в соответствии с выстроенным после сортировки порядком и записываются в результирующий файл.

Для распараллеливания данной операции в системе активного хранения была реализована следующая схема (рис. 4):

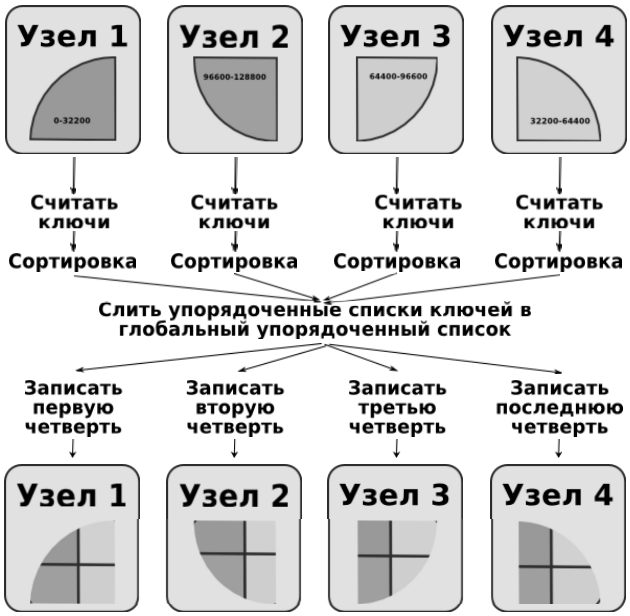

Рис. 4. Сортировка сейсмических данных

- На первом шаге производится считывание значений выбранных ключей для каждого из фрагментов исходного файла на том узле, где он расположен. На этом шаге следует ввести сквозную нумерацию по всем обрабатываемым фрагментам и связать с каждым ключом уникальный номер.
- Выполняется локальная сортировка считанных подмножества на соответствующих узлах хранения.
- Производится слияние упорядоченных подмножеств в единое отсортированное множество. В связи с линейной сложностью операции слияния упорядоченных списков данную операцию не следует распределять по всем доступным узлам. Данная операция может быть выполнена на одном или нескольких узлах в зависимости от размера исходного файла.
- На заключительном шаге выполняется запись фрагментов упорядоченного множества на узлах хранения. Для определения, к какому исходному файлу относилась рассматриваемая трасса, был использован уникальный номер заданный на этапе сквозной нумерации. Отсортированное множество трасс записывается таким образом, чтобы на i-м узле кластера были сохранены трассы с *(i-1)*size/N* по *i*size/N*, где N – количество доступных узлов, а *size* — общее количество трасс в исходном файле с данными.

Таким образом, после операции сортировки на каждом узле расположен фрагмент, содержащий *size/N* трасс, что позволит проводить дальнейшие операции обработки над каждым фрагментом по отдельности на узлах хранения. При этом конкатенация всех фрагментов позволит получить полностью отсортированный файл. Для реализации первого, второго и

последнего шага из приведенной схемы был использован шаблон параллельного программирования Map [5], входящий в состав библиотеки TSim.

Предварительное тестирование полученной операции сортировки в системе активного хранения данных применительно к файлу размером 767 Мб продемонстрировало следующие результаты:

- Последовательное выполнение: 1 мин 10 с
- 2 узла (данные разбиты на 2 фрагмента): 38 с
- 7 узлов (данные разбиты на 7 фрагментов): 14 с

5. Надежность и отказоустойчивость

Одним из ключевых требований при операциях с сейсмическими данными является надежность и отказоустойчивость системы. Так как обработка данных современных сейсмических наблюдений занимает значительное время [1], то выход из строя элементов вычислительного оборудования приводит к значительным экономическим потерям из-за несоблюдения сроков выполнения работ.

Для системы активного хранения разработан специальный модифицированный планировщик с функцией распознавания отказа одного или нескольких узлов. Для того чтобы своевременно распознать отказ узла планировщик с заданным интервалом времени рассылает активные сообщения, сообщающие о статусе отправки. Узел считается отказавшим, если на него не удается успешно передать сообщение либо очередную задачу.

Для продолжения работы системы после отказа одного из вычислительных узлов в TSim существует стандартный тип «задача с сохранением». В случае отказа какого-либо вычислительного узла, все назначенные на него и не завершенные на момент отказа задачи распределяются между оставшимися узлами в соответствии с алгоритмом планирования.

Отличительной особенностью системы активного хранения является расположение устройств хранения на вычислительных узлах. Соответственно, при отказе узла данные, расположенные на этом узле, становятся недоступны для дальнейшей обработки. Следовательно, сохранение только задач, без входных данных, не имеет смысла. С целью поддержки механизмов отказоустойчивости на уровне задач, создан специализированный класс C++ TaskSaved.

При создании задачи типа TaskSaved планировщик не только определяет, на каком узле расположены входные данные для указанной задачи и отправляет задачу на выявленный узел, но и выполняет копирование входных данных на устройство хранения, расположенное на другом узле. В случае отказа одного из вычислительных узлов в системе незавершенные задачи перенаправляются на тот узел, где расположена сохраненная копия. Такой подход позволяет эффективно организовать отказоустойчивые вычисления в системе активного хранения данных, но он связан с неизбежно высокими накладными расходами на создание и поддержание копий. Копия может удаляться как по завершении вычислительной задачи, с которой она связана, так и после завершения вычисления в целом.

Мы проводили теоретические исследования, нацеленные на минимизацию накладных расходов при организации отказоустойчивых вычислений [6]. Мы ведем работы по расширению данных исследований в приложении к обработке сейсмических данных. Естественной точкой для повышения надежности системы представляется сортировка сейсмических дан-

ных, которая приводит к дублированию входных данных, только в отсортированном формате. Сложность заключается в том, что исходные и отсортированные фрагменты имеют разное расположение трасс относительно вычислительных узлов.

6. Обзор аналогов

Наиболее близким аналогом является проект Seismic Hadoop, предложенный в начале этого года [7]. В рамках указанного проекта была проведена интеграция пакета Seismic Un*x и библиотеки Java Crunch, нацеленной на организацию MapReduce конвейеров. Seismic Hadoop использует вычислительный кластер Hadoop для хранения и обработки данных. Hadoop обеспечивает автоматическое резервирование данных и оптимизирован для выполнения Map Reduce операций.

Среди отличительных особенностей нашего подхода можно выделить:

1. Для организации распределенных вычислений используется библиотека шаблонных классов C++ TSim, что предоставляет нам возможности:
   - Быстрой смены стратегии планирования, разработки собственной стратегии в виде шаблонного параметра планировщика.
   - В классических случаях возможно использовать шаблоны распараллеливания Map и Reduce. При необходимости реализации более сложных схем распараллеливания возможно использование параллелизма по задачам, неготовых переменных в качестве примитивов синхронизации, активных сообщений и др.
   - Использование всех возможностей шаблонов C++, легкая интеграция с исходными кодами библиотеки Seismic Un*x, написанной на C.
2. В качестве уровня хранения в нашей системе используется кластерная файловая система Lustre, которая установлена на ведущих суперкомпьютерах мира, таких как K computer (модифицированная версия ФС Lustre, называемая FEFS), Tianhe-1A, Jaguar (модифицированная версия – Spider), TSUBAME 2.0 и др. Использование Lustre обеспечивает высокую производительность операций с данными и высокий потенциал масштабируемости.

Приведенные отличительные черты позволяют предположить, что интеграция системы активного хранения данных с пакетом Seismic Un*x может позволить исследовать эффективность различных комбинаций алгоритмов планирования и реплицирования данных и реализовать эффективные схемы распараллеливания для сложных подпрограмм обработки сейсмических данных.

Заключение

В работе предложен подход для организации распределенной обработки сейсмических данных на базе свободно распространяемого пакета Seismic Un*x и системы активного хранения данных с использованием TSim и ФС Lustre.

Несмотря на то, что работы по интеграции предложенных систем еще не завершены, на данном этапе рассмотрены ключевые проблемы обработки сейсмических данных, и для каждой предложено и обосновано использование соответствующего инструмента из арсенала системы активного хранения данных. Было проведено исследование производительности

разработанного программного прототипа по обработке сейсмических данных в системе активного хранения для оценки перспектив интеграции. Результаты эксперимента демонстрируют повышение производительности при использовании архитектуры активного хранения данных как по сравнению с последовательной обработкой, так и с распределенной обработкой без привязки к местам расположения данных.

В качестве продолжения исследований планируется провести тестирование производительности на реальных данных размером от нескольких сотен гигабайтов до нескольких терабайтов. В дальнейшем планируется разработка и реализация эффективных схем обеспечения надежности и отказоустойчивости.

Работа проводилась при финансовой поддержке Программы фундаментальных исследований Президиума РАН № 14). Работа проводилась при финансовой поддержке Министерства образования и науки Российской Федерации, Государственный контракт № 07.514.12.4007).

Литература

1. Курин, Е.А. Сейсморазведка и суперкомпьютеры / Е.А. Курин // Вычислительные методы и программирование. – 2011. – № 1. – С. 38–43.
2. Camp, W.J. Trends for high-performance scientific computing / W.J. Camp, P. Thierry // The Leading Edge. – 2010. – P. 44-–47.
3. Stockwell, J. Домашняя страница библиотеки Seismic Un*x / J. Stockwell. – URL: `http://www.cwp.mines.edu/cwpcodes/` (дата обращения: 28.12.2012).
4. Распределенный архив изображений дистанционного зондирования Земли / А.А. Московский, А.Ю. Первин, Е.О. Тютляева, Е.В. Шевчук // Программные продукты и системы. – 2009. – № 2(86). – С. 40–42.
5. Московский, А.А. Параллельная реализация модели map-reduce с использованием шаблонных классов C++ / А.А. Московский, А.Ю. Первин, Е.О. Тютляева // Программные продукты и системы. – 2009. – № 2(86). – С. 53–57.
6. Tyutlyaeva, E.O. An Initial Approximation to the Resource-Optimal Checkpoint Interval / E.O. Tyutlyaeva, A.A. Moskovsky // Parallel Computing Technologies, Lecture Notes in Computer Science. – 2011. – V. 6873/2011. – P. 384–389.
7. Wills, J. Seismic Data Science: Reflection Seismology and Hadoop / J. Wills. – URL: `http://www.cloudera.com/blog/2012/01/seismic-data-science-hadoop-use-case/` (дата обращения: 18.12.2012).

Екатерина Олеговна Тютляева, аспирант, Институт программных систем им. А.К. Айламазяна РАН, Исследовательский центр мультрипроцессорных систем (г. Переславль-Залесский, Российская Федерация), `ordi@xgl.pereslavl.ru`.

Александр Александрович Московский, кандидат химических наук, генеральный директор ЗАО «РСК Технологии» (г. Москва, Российская Федерация), `moskov@rsc-tech.ru`.

Сергей Станиславович Конюхов, кандидат химических наук, инженер, ЗАО «РСК Технологии» (г. Москва, Российская Федерация), `s.konyuhov@rsc-tech.ru`.

Евгений Александрович Курин, генеральный директор ООО «ГЕОЛАБ» (г. Москва, Российская Федерация), ekurin@yahoo.com.

APPLYING ACTIVE STORAGE APPROACH FOR SEISMIC DATA PROCESSING TASKS

E.O. Tyutlyaeva, Program System Institute of RAS (Pereslavl-Zalessky, Russian Federation)**,**
A.A. Moskovsky, RSK Tech (Moscow, Russian Federation)**,**
S.S. Konuhov, RSK Tech (Moscow, Russian Federation)**,**
E.A. Kurin OOO «GEOLAB» (Moscow, Russian Federation)

In this paper, we propose an approach to distributed seismic data processing using active storage system based on TSim C++ Library and Lustre file System and open-source package Seismic Un*x. Data processing using storage nodes demonstrates dramatic performance increase by avoiding redundant data transfers over networks, interconnects, or storage busses. Performance tests of developed programming prototype of seismic data processing in active storage system allows to evaluate integration prospects.

Keywords: active storages, seismic data processing, distributed storing and processing data.

References

1. Kurin E.A. Seismic Prospecting and Supercomputers // *Numerical methods and programming.* 2011. Vol. 12, No. 1. P. 38–43.
2. Camp W.J, Thierry P. Trends for High-Performance Scientific Computing // *The Leading Edge.* 2010. Vol. 29. P. 44—47.
3. Stockwell J. Seismic Un*x Home Page. URL: *http://www.cwp.mines.edu/cwpcodes/* (accessed: 28.12.2012)
4. Tyutlyaeva E.O., Moskovsky A.A., Pervin A.Y., Schevchuk E.V. Distributed Remote Sensing Data Storage // *Software and Systems.* 2009. Vol. 2, No. 86. P. 40–42.
5. Tyutlyaeva E.O., Moskovsky A.A., Pervin A.U. Distributed Implementation of Map-Reduce Model Using C++ Template Classes // *Software and Systems.* 2009. Vol. 2, No. 86. P. 53–57.
6. Tyutlyaeva E.O., Moskovsky A.A. An Initial Approximation to the Resource-Optimal Checkpoint Interval // *Lecture Notes in Computer Science.* 2011. Vol. 6873/2011. P. 384–389.
7. Wills, J. Seismic Data Science: Reflection Seismology and Hadoop URL: *http://www.cloudera.com/blog/2012/01/seismic-data-science-hadoop-use-case/* (accessed: 28.12.2012).

Поступила в редакцию 23 ноября 2012 г.

Краткие сообщения

УДК 519.863

О НЕКОТОРЫХ СВОЙСТВАХ N-ПОСЛЕДОВАТЕЛЬНОСВЯЗНОЙ ЦЕПИ

Р.Э. Шангин

Вводится класс n-последовательносвязных цепей. Рассматриваются области применения n-последовательносвязных цепей, в частности задачи оптимального размещения в дискретных постановках и задачи выбора оптимального поведения в системах, описываемых управляемыми марковскими процессами. Приводятся основные характеристики n-последовательносвязной цепи, такие как число ребер, размер максимальной клики, хроматическое и цикломатическое число и др. Исследуются свойства n-последовательносвязных цепей. Определяются отношения класса n-последовательносвязных цепей к классам совершенных, триангулированных, полных и расщепляемых графов.

Ключевые слова: триангулированный граф, хордальный граф, дерево клик, задача Вебера, n-последовательносвязная цепь.

Введение

Пусть $G = (J, E)$ – неориентированный граф с множеством вершин J и множеством ребер E. Пусть $N(j)$ – множество вершин графа $G = (J, E)$, смежных с вершиной j. Пусть $\varphi(G)$ – плотность графа G. Далее будем предполагать, что на множестве вершин J введена нумерация и каждая вершина отождествлена с присвоенным ей номером.

Определение 1. *Связный граф $G = (J, E)$ называется n-последовательносвязной цепью (n-sequentially connected chain), если на множестве его вершин можно задать такую нумерацию, что для любой вершины графа G с номером j, имеет место включение*

$$N(j) \subseteq \{(j-n), ..., (j-1), (j+1), ..., (j+n)\} : n = \varphi(G) - 1.$$

В дальнейшем такие вершины i и k будем именовать крайними вершинами n-последовательносвязной цепи. Заметим, что такие вершины i и k будут симплициальные.

На рис. 1 для различных n представлены n-последовательносвязные цепи.

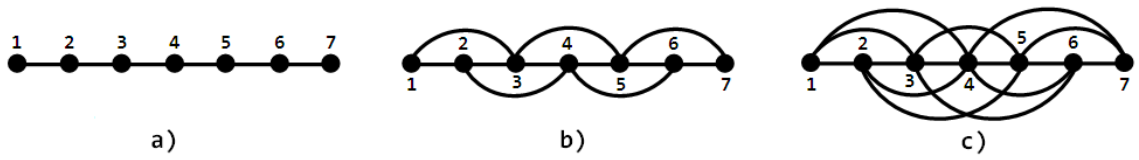

Рис. 1. n-последовательносвязные цепи: a) 1-последовательносвязная цепь; b) 2-последовательносвязная цепь; c) 3-последовательносвязная цепь

То есть 1-последовательносвязная цепь представляет собой простую цепь, т.к. для любых j $|N(j)| \leq 2$ и каждая вершина такой цепи связана не более чем с одной предшествующей вершиной. В 2(3)-последовательносвязной цепи каждая вершина связана не более чем с двумя (тремя) предыдущими вершинами соответственно.

На данный момент можно выделить два основных направления использования n-последовательносвязных цепей.

Первое – задачи размещения, в частности, задача Вебера в дискретной постановке [1], так как структура некоторых производственных процессов представляет собой n-последовательносвязную цепь. Например, в работе [2] предложен точный квазиполиномиальный алгоритм для решения задачи Вебера для неориентированной n-последовательносвязной цепи и конечного дискретного множества мест размещения.

Второе направление – решение задач выбора оптимального поведения в системах, описываемых управляемыми марковскими процессами, когда цепь Маркова является n-сложной, так как структура n-сложной марковской цепи представляет собой n-последовательносвязную цепь.

Основные характеристики и свойства n-последовательносвязной цепи

Приведем характеристики n-последовательносвязной цепи $G = (J, E)$.

1. Число ребер графа G равно:

$$|E| = (|J| - n) \cdot n + \sum_{i=\overline{1,n}} (n - i) = (|J| - n) \cdot n + \frac{n^2 - n}{2} = n \cdot \left(|J| - \left(\frac{n+1}{2} \right) \right);$$

2. Размер максимальной клики графа G равен $n + 1$, причем существует $|J| - n$ таких максимальных клик;
3. Граф G имеет 2 симплициальные вершины;
4. Число вершинной связности ξ_G графа G равно n, т.е. любая n-последовательно-связная цепь является n-связным графом;
5. Хроматическое число χ_G графа G равно $n + 1$;
6. Цикломатическое число C_G графа G равно:

$$C_G = n \cdot \left(|J| - \left(\frac{n+1}{2} \right) \right) - |J| + 1 = \left(|J| - \frac{n}{2} - 1 \right) \cdot (n - 1).$$

Рассмотрим некоторые свойства n-последовательносвязной цепи $G = (J, E)$.

Теорема 1. *В n-последовательносвязной цепи $G = (J, E)$ ни один из порожденных подграфов не является простым циклом длины $l \geq 4$.*

Доказательство. В пронумерованной n-последовательносвязной цепи $G = (J, E)$ рассмотрим две цепи $L_1 = (J_{L1}, E_{L1})$ и $L_2 = (J_{L2}, E_{L2})$, номера вершин которых принадлежат множествам N_{L1} и N_{L2} соответственно, причем

$$N_{L1} = \{j, j + k_1, j + k_1 + k_2, ..., j + \sum_{i=1}^{m} k_i\},$$

где $j \in \{1, 2, ..., |J_{L1}|\}$ и для любых $i = 1, 2, ..., m$ целые числа $k_i \in \{1, 2, ..., n\}$, при том, что $j + \sum_{i=1}^{m} k_i \leq |J_{L1}|$;

$$N_{L2} = \{j'', j'' - h_1, j'' - h_1 - h_2, ..., j'' - \sum_{t=1}^{w} h_t, j\},$$

где $j'' = j + \sum_{i=\overline{1,m}} k_i$ и для любых $t = 1, 2, ..., w$ целые числа $h_t \in \{1, 2, ..., n\}$, при том, что $j'' - \sum_{t=1}^{w} h_t > j$. Пусть $L = L_1 \cup L_2$ – простой цикл длины $m + w + 1 \geq 4$.

Пусть некоторая вершина цепи L_2 с номером $j'' - \sum_{t=1}^{\varpi} h_t$, где $\varpi \in \{1, 2, ..., w\}$, находится «между» вершинами цепи L_1 с номерами $j + \sum_{i=1}^{\rho} k_i$ и $j + \sum_{i=1}^{\rho+1} k_i$, где $\rho \in \{1, 2, ..., m\}$ и $k_{\rho+1} \geq 2$. Очевидно, что

$$j + \sum_{i=1}^{\rho} k_i < j'' - \sum_{t=1}^{\varpi} h_t < j + \sum_{i=1}^{\rho+1} k_i.$$

Для такой вершины цепи L_2 с номером $j'' - \sum_{t=1}^{\varpi} h_t$ найдется хотя бы одно ребро (хорда) $(j + \sum_{i=1}^{\rho} k_i, j'' - \sum_{t=1}^{\varpi} h_t) \in E$ либо $(j'' - \sum_{t=1}^{\varpi} h_t, j + \sum_{i=1}^{\rho+1} k_i) \in E$, которое не принадлежит множествам ребер цепей L_1 и L_2, т. к. цикл $L = L_1 \cup L_2$ по определению простой и

$$(\forall j \in \{1, 2, ..., |J|\}) \qquad N(j) \subseteq \{(j - n), .., (j - 1), (j + 1), .., (j + n)\},$$

и для любых $i \in \{1, 2, ..., m\}$ и $t \in \{1, 2, ..., w\}$ справедливо неравенство $1 \leq k_i, h_t \leq n$. Исходя из этого, в графе G ни один из порожденных подграфов не является простым циклом длины $l \geq 4$. Теорема 1 доказана. □

Использованные в доказательстве теоремы 1 простые цепи L_1 и L_2, а так же хорды, соединяющие несмежные вершины простого цикла L в 3-последовательносвязной цепи представлены на рис. 2.

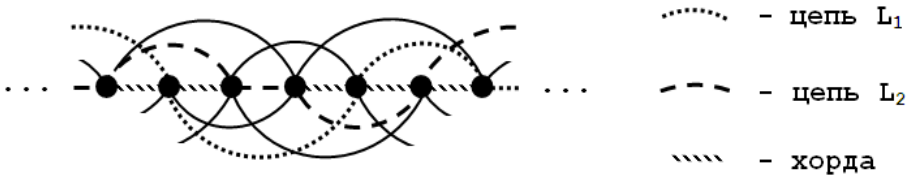

Рис. 2. bsd

Исходя из теоремы 1 n-последовательносвязная цепь при любых значениях параметра n является триангулированным (хордальным) графом [3–6]. Несмотря на справедливость теоремы 1 имеют место следующие свойства n-последовательносвязной цепи.

Теорема 2. *В n-последовательносвязной цепи $G = (J, E)$ при $n > 2$ с количеством вершин $|J| \geq 2n + 1$ существует порожденный подграф, являющийся циклом длины $l \geq 4$.*

Доказательство. Для доказательства теоремы 2 необходимо и достаточно найти в графе G такой цикл L длины $l \geq 4$, для которого в графе G не существует ребра соединяющего две несмежные вершины цикла L.

Пусть $L = (J_L, E_L) : |E_L| \geq 4$ цикл в графе G, номера вершин которого принадлежат множеству N_L, причем

$$N_L = \{j, j + n, j + 2n, ..., j + kn, j + kn - 1, j + (k - 1)n, j + (k - 1)n - 1, j + (k - 2)n, ..., j\},$$

при условии, что целое число $k = 2, 3, ...$, при том, что $j + kn \leq |J_L|$.

Очевидно, что для любой вершины $l \in J_L$ не существует таких ребер $(l, m) \in E : m \in J_L$, которые бы не принадлежали множеству ребер E_L цикла L, так как известно, что для любых $j \in \{1, 2, ..., |J|\}$ множество $N(j) \subseteq \{(j-n), .., (j-1), (j+1), .., (j+n)\}$. Теорема 2 доказана. □

Такой цикл L в n-последовательносвязной цепи при $n = 3, 4$ представлен на рис. 3.

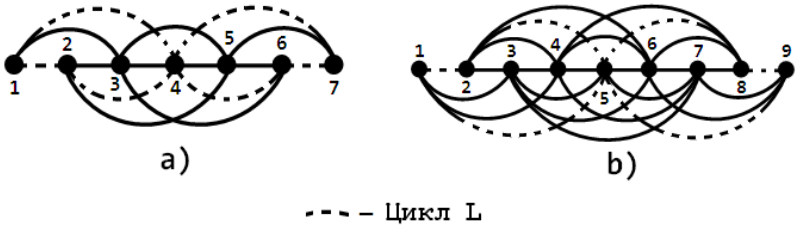

Рис. 3. csd

Теорема 3. *В любой n-последовательносвязной цепи $G = (J, E)$ при $n \geq 4$ существует порожденный подграф, являющийся циклом длины $l \geq 4$.*

Доказательство. Положим, что $A = (J_A, E_A)$ – подграф n-последовательносвязной цепи $G = (J, E)$ при $n \geq 4$, образующий клику размера $k + 1$, где $4 \leq k \leq n : k(\text{mod } 2) \equiv 2$.

Так как степени всех вершин такой клики A четны, то подграф M индуцированный множеством вершин клики A является эйлеровым графом, а значит найдется такой цикл $L = M$, который включает в себя все вершины и ребра порожденного графа $M \in G$.

Исходя из этого, в графе G существует порожденный подграф L, являющийся циклом длины $l \geq 4$. Теорема 3 доказана. □

Несмотря на справедливость теорем 2 и 3 имеет место следующая теорема.

Теорема 4. *В n-последовательносвязной цепи $G = (J, E)$ при $n = 2$ ни один из порожденных подграфов не является циклом длины $l \geq 4$.*

Доказательство. Пусть $L = (J_L, E_L) : |E_L| \geq 4$ – цикл в графе G, где $J_L = \{j, i_1, i_2, ..., i_t, j\} : j \in J$.

Рассмотрим два случая цикла L: первый случай – когда вершина i_1 есть вершина с номером $j + 1$, второй случай – когда вершина i_1 есть вершина с номером $j + 2$.

Рассмотрим первый вариант первого случая цикла L, когда

$$J_L = \{j, j + 1, i_2, ..., i_r, j', j + 1, j - 1, i_{r+4}, ..., i_t, j\} : j' \in \{j + 2, j + 3\}.$$

В данном варианте при любых $j' \in \{j + 2, j + 3\}$ в графе G имеется ребро $(j, j + 2) \in E$, соединяющее две несмежные вершины цикла L (рис. 4, a – b), т.к. вершина с номером $j + 2$ при любых j содержится в цикле L и цикл заканчивается в вершине с номером j. В дальнейшем такое ребро будем называть хордой.

Во втором варианте первого случая цикла L вершина $i_t = j + 2$. Здесь, в свою очередь, возможны два случая: первый, когда $i_2 = j + 3$ – с хордой $(j + 1, j + 2) \in E$, т.к. для любых $i_{t-1} \in \{j + 3, j + 4\}$ ребро $(j + 1, j + 2) \in E$ не принадлежит циклу L (рис. 4, c), второй,

когда $i_2 = j + 2$ и $i_3 \in \{j + 3, j + 4\}$ – с хордой $(j + 1, j + 3) \in E$, т.к. при $i_3 = j + 3$ вершина $i_{t-1} = j + 4$, а при $i_3 = j + 4$ вершина $i_{t-1} = j + 3$ и во всех случаях ребро $(j + 1, j + 3) \in E$ не принадлежит циклу L (рис. 4, d – e).

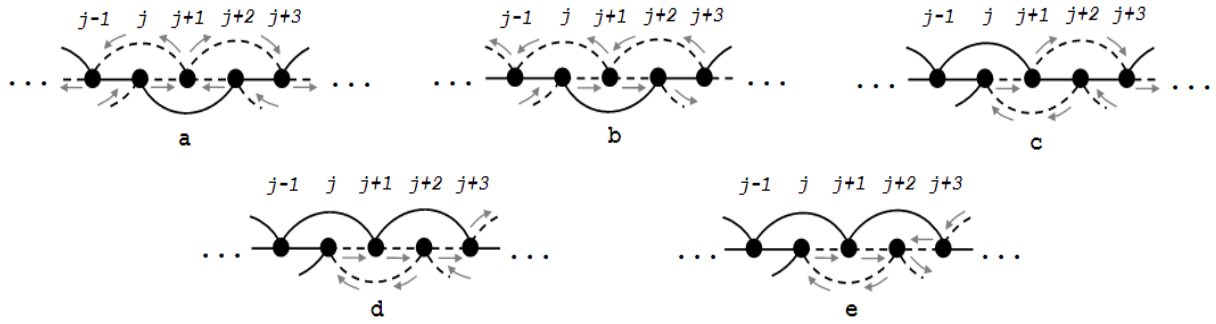

Рис. 4. Первый вариант построения цикла L в 2-последовательносвязной цепи

Рассмотрим первый вариант второго случая цикла L, когда

$$J_L = \{j, j + 2, i_2, ..., i_r, j + 3, j + 1, j', i_{r+4}, ..., i_t, j\} : j' \in \{j, j - 1\}.$$

В данном варианте при $j' = j - 1$ в графе G содержатся две хорды $(j, j + 1) \in E$ и $(j + 1, j + 2) \in E$ (рис. 5, a), а при $j' = j$ – хорда $(j + 1, j + 2) \in E$ (рис. 5, b).

Во втором варианте второго случая цикла L, когда

$$J_L = \{j, j + 2, i_2, ..., i_r, j + 2, j + 1, j', i_{r+4}, ..., i_t, j\} : j' \in \{j, j - 1\},$$

при $j' = j - 1$ в графе G содержится хорда $(j, j + 1) \in E$ (рис. 5, c), а при $j' = j$, когда $i_2 \in \{j + 3, j + 4\}$ – хорда $(j + 1, j + 3) \in E$ (рис. 5, d – e).

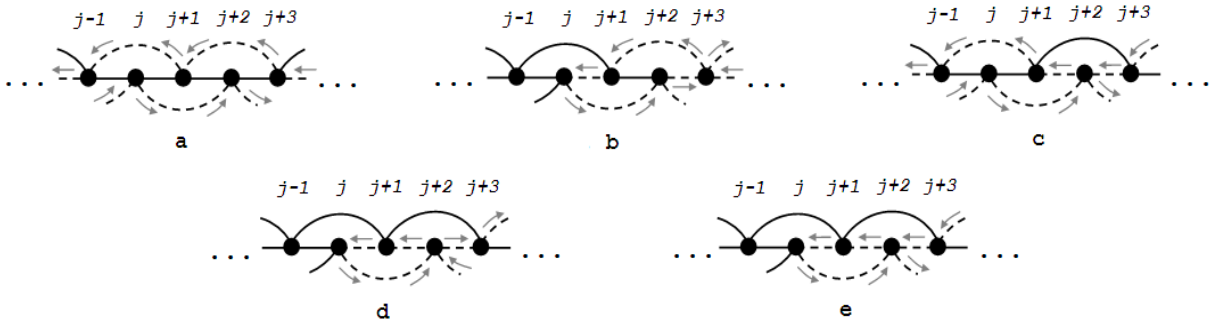

Рис. 5. Второй вариант построения цикла L в 2-последовательносвязной цепи

Так как цикл L во всех возможных случаях содержит хорду, то в n-последовательносвязной цепи $G = (J, E)$ при $n = 2$ ни один из порожденных подграфов не является циклом длины $l \geq 4$. Теорема 4 доказана. □

Так как согласно теореме 1 любая n-последовательносвязная цепь является триангулированным графом, свойства n-последовательносвязной цепи повторяют свойства хордального графа. В частности справедливы

Свойство 1. Любая n-последовательносвязная цепь является совершенным графом.

Свойство 2. Любое разделяющее множество вершин n-последовательносвязной цепи, минимальное относительно включения, есть клика.

Свойство 3. Если n-последовательносвязная цепь отлична от полного графа, то в ней имеется две симплициальные вершины.

Свойство 4. Каждая часть n-последовательносвязной цепи относительно разделяющего множества вершин, являющегося кликой – совершенный граф.

Свойство 5. Неориентированная n-последовательносвязная цепь имеет дерево клик, причем древовидная ширина n-последовательносвязной цепи равна $n + 1$.

Отношение класса n-последовательносвязных цепей к другим классам графов представлено графически на рисунке 6.

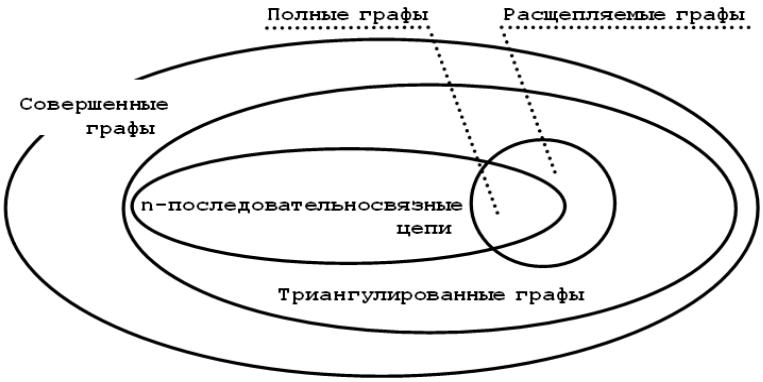

Рис. 6. Отношения классов графов

Т.е. класс n-последовательносвязных цепей принадлежит классам совершенных и триангулированных графов, при том, что n-последовательносвязные цепи с числом вершин $|J| = n+1$ принадлежат классу полных графов, а n-последовательносвязные цепи с числом вершин $|J| = n + 2$ – классу расщепляемых графов.

Заключение

В работе введено понятие неориентированной n-последовательносвязной цепи. Обозначены области применения n-последовательносвязных цепей. Рассмотрены основные характеристики n-последовательносвязной цепи.

Доказано, что в n-последовательносвязной цепи при любых значениях параметра n ни один из порожденных подграфов не является простым циклом длины $l \geq 4$, из чего следует, что класс n-последовательносвязных цепей принадлежит классу триангулированных графов.

Так же доказано, что в n-последовательносвязной цепи при $n \geq 2$ с количеством вершин $|J| \geq 2n + 1$ и в любой n-последовательносвязной цепи при $n \geq 4$ существует порожденный подграф, являющийся циклом длины $l \geq 4$, при том, что в n-последовательносвязной цепи при $n = 2$ ни один из порожденных подграфов не является циклом длины $l \geq 4$.

Определено отношение класса n-последовательносвязных цепей к классам совершенных, триангулированных, полных и расщепляемых графов.

Литература

1. Panyukov, A.V. Polynomial Algorithms to Finite Veber Problem for a Tree Network / A. V. Panyukov, B.V. Pelzwerger // Journal of Computational and Applied Mathematics. – 1991. – Vol. 35. – P. 291–296.
2. Шангин, Р.Э. Детерминированный алгоритм для решения задачи Вебера для n-последовательносвязной цепи / Р.Э. Шангин // XIII Всероссийская конференция молодых ученых по математическому моделированию и информационным технологиям: Труды Всероссийской конференции (Новосибирск, 15–17 октября 2012 г.). URL: http://conf.nsc.ru/ym2012/ru/reportview/137128 (дата обращения 15.10.2012).
3. Bender, E.A. Almost All Chordal Graphs Split / E.A. Bender, L.B. Richmond, N. C. Wormald // Journal of the Australian Mathematical Society. – 1985. – Vol. 38. – P. 214–221.
4. McKee, T.A. On the Chordality of a Graph / T.A. McKee, E.R. Scheinerman // Journal of Graph Theory. – 1993. – Vol. 17. – P. 221–232.
5. McKee, T.A. Beyond Chordal Graphs / T.A. McKee // Journal of Combinatorial Mathematics and Combinatorial Computing. – 1997. – Vol. 23. – P. 21–31.
6. Dirac, G.A. On Rigid Circuit Graphs / G.A. Dirac // Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg. – 1961. – Vol. 25. – P. 71–76.

Роман Эдуардович Шангин, аспирант, кафедра экономико-математических методов и статистики, Южно-Уральский государственный университет (г. Челябинск, Российская Федерация), shanginre@gmail.com.

ON SOME PROPERTIES OF N-SEQUENTIALLY CONNECTED CHAIN

R.E. Shangin, South Ural State University (Chelyabinsk, Russian Federation)

It is introduced the class of the nonoriented n-sequentially connected chain. It is considered the application fields of the n-sequentially connected chains, in particular the problems of optimal location in discrete formulations, and the problems of selection the optimal conduct in systems, described by Markov controllable processes. The main characteristics of n-sequentially connected chains, such as the number of edges, the size of the maximum clique, the chromatic and the cyclomatic numbers, etc. are given. The relations of the class n-sequentially connected chains to perfect, triangulated, composite and splittable classes of graphs are determined.

Keywords: triangulated graph, chordal graph, tree-clique, Weber problem, the n-sequen-tially connected chain.

References

1. Panyukov A.V., Pelzwerger B.V. Polynomial Algorithms to Finite Veber Problem for a Tree Network. Journal of Computational and Applied Mathematics. 1991. Vol. 35. P. 291–296.
2. Shangin R.E. Determinirovannyj algoritm dlja reshenija zadachi Vebera dlja n-posledovatelnosvjaznoj cepi [The Deterministic Algorithm for Solving the Weber Problem for

the n-sequentially Connected Chain]. XIII Vserossijskaja konferencija molodyh uchenyh po matematicheskomu modelirovaniju i informacionnym tehnologijam: trudy Vserossijskoj konferencii (Novosibirsk, Russia, 15–17 oktjabrja 2012) [The XIII All-Russian Conference of Young Scientists on Mathematical Modeling and Information Technologies: Proceedings of the All-Russian Conference (Novosibirsk, Russia, 15–17 October 2012)]. URL: `http://conf.nsc.ru/ym2012/ru/reportview/137128.pdf`.

3. Bender E.A. Almost All Chordal Graphs Split. Journal of the Australian Mathematical Society. 1985. Vol. 38. P. 214–221.
4. McKee T.A. On the Chordality of a Graph. Journal of Graph Theory. 1993. Vol. 17. P. 221–232.
5. McKee T.A. Beyond Chordal Graphs. Journal of Combinatorial Mathematics and Combinatorial Computing. 1997. Vol. 23. P. 21–31.
6. Dirac G.A. On Rigid Circuit Graphs. Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg. 1961. Vol. 25. P. 71–76.

Поступила в редакцию 1 ноября 2012 г.

УДК 004.043

ПОДХОД К ИНТЕГРАЦИИ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ В РЕЛЯЦИОННУЮ СУБД НА ОСНОВЕ ГЕНЕРАЦИИ ТЕКСТОВ ХРАНИМЫХ ПРОЦЕДУР

Т.В. Речкалов

Представлен подход к интеграции интеллектуального анализа данных (ИАД) в реляционную СУБД. Подход предполагает использование XML-разметки алгоритма ИАД, выраженного на языке SQL. Разметка позволяет выполнить автоматическую генерацию хранимых процедур на языке SQL, реализующих данный алгоритм, в зависимости от специфицированных пользователем таблиц исходных данных и параметров алгоритма. Приведено описание предложенного языка разметки. Если для решения задачи ИАД имеется несколько алгоритмов, подход предполагает генерацию SQL-кода, реализующего наиболее эффективный из них. Выбор наиболее эффективного алгоритма осуществляется на основе использования имеющейся в современных СУБД команды EXPLAIN, позволяющей получить оценку времени исполнения запроса SQL без его фактического выполнения. Описана модульная структура и интерфейс программной системы, реализующей данный подход.

Ключевые слова: интеллектуальный анализ данных, реляционная СУБД, SQL.

Введение

Интеграция алгоритмов интеллектуального анализа данных (ИАД) в реляционные системы управления базами данных (СУБД) представляет собой одну из актуальных задач аналитической обработки данных [1]. Это обусловлено тем, что существующие алгоритмы ИАД как правило предполагают размещение анализируемых данных в оперативной памяти (см. например [2, 3]), и их совместное использование с СУБД требует значительных накладных расходов, связанных с экспортом исходных данных задачи во внешнюю аналитическую утилиту и импортом результатов работы обратно в базу данных. Реализация алгоритмов ИАД на языке SQL позволяет избежать накладных расходов на экспорт-импорт данных и обеспечивает анализ данных, не помещающихся в оперативную память, на основе стандартных механизмов СУБД.

Исследования по реализации алгоритмов ИАД на языке SQL представлены следующими работами. Известны алгоритмы решения задачи рыночной корзины SETM [4] и ScanOnce [5]. Реализация алгоритма кластеризации K-Means на языке SQL рассмотрена в [1]. Реализация алгоритма нечеткой кластеризации Fuzzy c-Means описана в [6, 7]. Фреймворк для проведения ИАД в СУБД представлен в работе [8].

В данной работе описаны предварительные результаты по разработке подхода к интеграции ИАД в реляционную СУБД на основе использования XML-разметки алгоритма ИАД, выраженного на языке SQL. Разметка позволяет выполнить автоматическую генерацию хранимых процедур на языке SQL, реализующих данный алгоритм, в зависимости от указанных пользователем таблиц исходных данных и параметров алгоритма.

Работа организована следующим образом. В первом разделе на примере задачи рыночной корзины описан подход к автоматизированной генерации хранимых процедур ИАД на основе использования языка разметки алгоритма, а так же описан механизм выбора наиболее эффективного алгоритма ИАД для заданных исходных данных. Во втором разделе описана программная реализация предложенного подхода. В заключении представлена сводка основных результатов сообщения.

1. Автоматизированная генерация текстов хранимых процедур для интеллектуального анализа данных

В данном разделе описан подход к интеграции ИАД и реляционной СУБД на основе автоматизированной генерации текстов хранимых процедур на языке SQL, реализующих алгоритмы ИАД. Изложение проводится на примере алгоритма SETM решения задачи рыночной корзины [9]. Задача *рыночной корзины (market-basket problem)* заключается в нахождении всех наборов (множеств) товаров, которые часто приобретаются совместно и формально определяется следующим образом.

Пусть даны множество товаров $I=\{i_1, i_2, \ldots, i_m\}$, множество транзакций D, где каждая транзакция представляет собой множество товаров T **из** I. Каждая транзакция имеет уникальный идентификатор *TID*.

Поддержка (support) набора товаров X определяется как количество транзакций из D, каждая из которых содержит данный набор товаров X. Пусть *minsup* – минимальное значение поддержки, при котором набор товаров считается часто встречающимся.

Обозначим за F *(frequent itemsets)* множество часто встречающихся наборов. Множество F_k содержит все возможные наборы товаров из k элементов, поддержка которых не меньше *minsup*. Решением задачи является множество $F=\bigcup_k F_k$. Пример задачи рыночной корзины и ее решения приведены на рис. 1.

TID	Items
10	{A, C, D}
20	{B, C, E}
30	{A, B, C, E}
40	{B, E}

а) исходные данные

k	F_k	Support
1	{A}	2
	{B}, {C}, {E}	3
2	{A, C}, {B, C}, {B, E}, {C, E}	2
3	{B, C, E}	2

б) решение

Рис. 1. Пример задачи рыночной корзины *(minsup=2)*

При реализации алгоритма ИАД в рамках реляционной СУБД исходные данные и решение задачи должны быть представлены в виде реляционных таблиц, а алгоритм записан на языке запросов SQL. На рис. 2 представлены преобразованные исходные данные и решение задачи рыночной корзины из рис. 1.

TID	Item
10	A
10	C
10	D
...	...
40	E

а) исходные данные

F1

item1	Support
A	2
B	3
C	3
E	2

F2

item1	item2	Support
A	C	2
B	C	2
B	E	3
C	E	2

F3

item1	item2	item3	Support
B	C	E	2

б) решение

Рис. 2. Преобразование данных задачи рыночной корзины к реляционному формату

Идея алгоритма заключается в итеративной генерации множества C_k кандидатов в частые наборы и последующем отборе кандидатов с подходящим значением поддержки множества F_k. Итерации алгоритма являются рекуррентными. Пример текста запросов для итераций 2 и 3 приведен на рис. 3, изменяющиеся части запросов подчеркнуты.

```
-- Генерация кандидатов в частые
-- 2-наборы
INSERT INTO C2 as
SELECT p.tid, p.item1,
q.item1 as item2
FROM F1 p, F1 q
WHERE q.tid = p.tid
and q.item1 > p.item1;
-- Нахождение частых 2-наборов
SELECT p.item1, p.item2, COUNT(*) as
support
FROM C2 p
GROUP BY p.item1, p.item2
HAVING COUNT(*) >= minsup;
```

а) k=2

```
-- Генерация кандидатов в частые
-- 3-наборы
INSERT INTO C3 as
SELECT p.tid, p.item1, p.item2,
q.item1 as item3
FROM F2 p, F1 q
WHERE q.tid = p.tid
and q.item1 > p.item2;
-- Нахождение частых 3-наборов
SELECT p.item1, p.item2, p.item3, COUNT(*)
as support
FROM C3 p
GROUP BY p.item1, p.item2, p.item3
HAVING COUNT(*) >= minsup;
```

б) k=3

Рис. 3. Фрагмент алгоритма SETM решения задачи рыночной корзины

Заметим, что в текстах запросов для различных итераций меняются только имена используемых таблиц и списки столбцов, а структура запросов остается прежней.

В соответствии с этим нами предлагается язык разметки SQL-алгоритмов ИАД, названный *AEML (Algorithm Explain Markup Language)*. С помощью AEML можно описать строки запросов SQL, зависящие от входных данных. Описание на языке AEML преобразовывается в текст хранимой процедуры на языке SQL.

На рис. 4 представлен пример *AEML* разметки для запросов, приведенных на рис. 3. Язык AEML имеет следующие основные теги.

Тег `<text>` вставляет заключенное в нем содержимое без изменений. Тег `<list>` используется для генерации списков значений с разделителями. Например, списков колонок в запросе или значений для вставки в таблицу (рис. 3, строки 5, 15). Тег `<listParam>` используется для получения элементов списка (рис. 3, строки 6, 15). Например, в списке перечисляются значения для вставки в таблицу.

Тег `<loop>` используется для генерации наборов строк. Например, для генерации набора запросов в зависимости от параметра (рис. 3, строка 1). Тег `<loopParam>` используется для получения переменных цикла. Тег `<externalParam>` используется для разметки

входных параметров алгоритма. Например, значение минимальной поддержки (рис. 3, строка 23).

Тег `<internalParam>` используется для разметки внутренних параметров алгоритма, значение которых определятся в процессе генерации текста алгоритма. Внутренние параметры могут быть производными от внешних параметров, (например, значение минимальной поддержки, увеличенное на единицу) или от контекста генерации (например, имя таблицы, зависящее от номера итерации алгоритма (строки 2 и 9 на рис. 3).

```
1.  <loop name=‘‘iterator_K’’>
2.    <text> INSERT INTO </text> <internalParam name=‘‘candidateTableName’’/>
3.    <text> as </text>
4.    <text> SELECT p.tid, </text>
5.    <list separator=‘‘,’’ index=‘‘1’’ generator=‘‘colList’’>
6.      <text>p.item</text> <listParam index=‘‘1’’/>
7.    </list>
8.    <text> q item1 as item </text> <internalParam name=’’newItemColumnIndex’’/>
9.    <text> FROM </text> <internalParam name=‘‘freqItemsetTableName’’/>
10.   <text> p, F1 q </text>
11.   <text> WHERE q.tid = p.tid and
12.     q.item1 &gt; p.item </text> <internalParam name=‘‘itemNum’’/> <text>;</text>
13.   <text> SELECT </text>
14.   <list separator=‘‘,’’ index=‘‘1’’ generator=‘‘colList’’>
15.     <text>p.item</text> <listParam index=‘‘1’’/>
16.   </list>
17.   <text>, COUNT(*) as support</text>
18.   <text> FROM </text> <internalParam name=‘‘freqItemsetTableName’’/> <text> p</text>
19.   <text> GROUP BY </text>
20.   <list separator=‘‘,’’ index=‘‘1’’ generator=‘‘colList’’>
21.     <text>p.item</text> <listParam index=‘‘1’’/>
22.   </list>
23.   <text>HAVING COUNT(*) &gt;= </text> <externalParam name=’’minsup’’/>
24.   <text>;</text>
25. </loop>
```

Рис. 4. Разметка запросов, приведенных на рис. 3

Помимо алгоритма SETM, существуют другие алгоритмы решения задачи рыночной корзины, например, алгоритм ScanOnce [5]. Если для решения задачи ИАД имеется несколько алгоритмов, осуществляется генерация SQL-кода для наиболее эффективного из них.

Выбор наиболее эффективного алгоритма выполняется на основе использования утилиты EXPLAIN [10], входящей в состав современных СУБД. EXPLAIN позволяет получить оценку времени исполнения запроса SQL без его фактического выполнения. Утилита, используя технику оптимизации запросов, строит наиболее эффективный план запроса и выдает оценку времени выполнения в условных единицах.

Оценка алгоритма ИАД производится путем суммирования оценок EXPLAIN для всех запросов, входящих в реализацию алгоритма. Алгоритм с наименьшей оценкой является предпочтительным.

2. Программная реализация подхода

Подход, описанный в разделе 1, реализован в виде сторонней по отношению к СУБД утилиты, названной *DM-EXPLAIN*. Варианты использования утилиты DM-EXPLAIN приведены на рис. 5.

Администратор базы данных осуществляет добавление в систему описания алгоритма ИАД на языке AEML. Программист с помощью системы выполняет генерацию хранимой процедуры на языке SQL, реализующей этот алгоритм, в соответствии с заданными параметрами. В процессе генерации используются результаты оценки времени выполнения запросов, получаемых с помощью стандартной утилиты EXPLAIN.

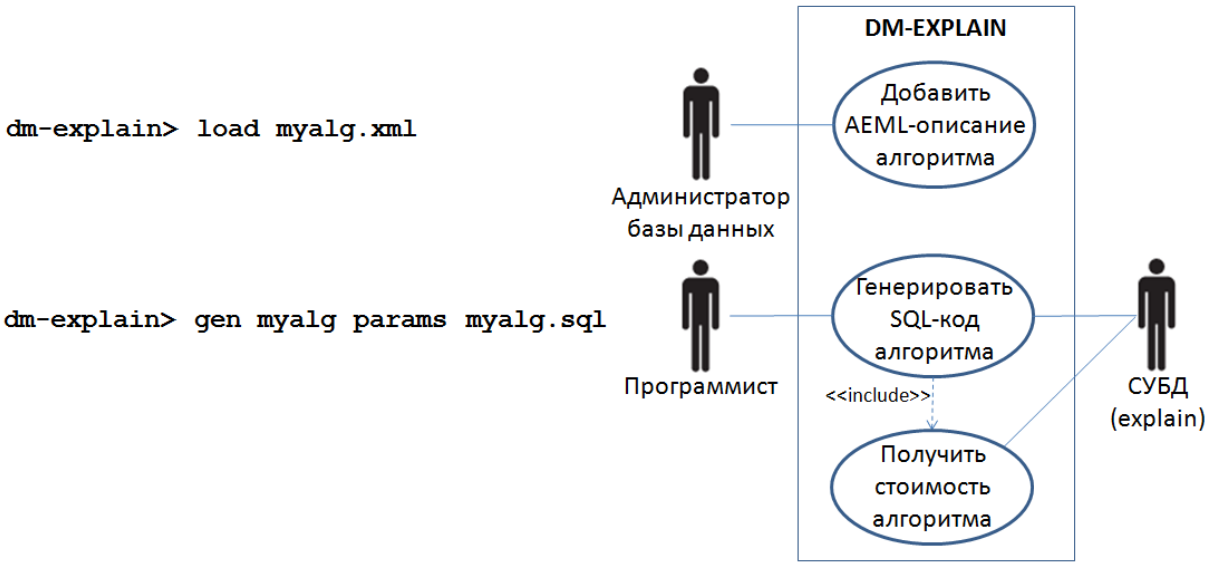

Рис. 5. Диаграмма вариантов использования утилиты DM-EXPLAIN

Модульная структура утилиты DM-EXPLAIN представлена на рис. 6.

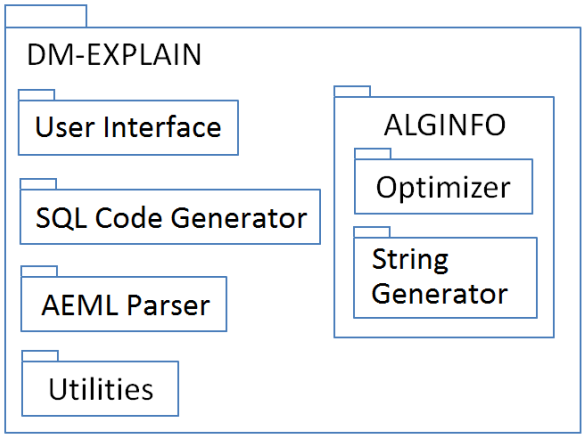

Рис. 6. Диаграмма пакетов утилиты DM-EXPLAIN

Пакет User Interface реализует пользовательский интерфейс утилиты.

Пакет SQL Code Generator обеспечивает функции автоматической генерации текстов хранимых процедур на языке SQL.

Пакет AEML Parser отвечает за разбор AEML разметки.

Пакет ALGINFO представляет собой API для разработки алгоритмов и состоит из двух вложенных пакетов: Optimizer и String Generator. Пакет Optimizer предоставляет

API для оценки сложности алгоритмов. Пакет String Generator предоставляет API для генерации подстановочных значений алгоритмов.

Пакет Utilities содержит классы, отвечающие за взаимодействие утилиты с СУБД.

Заключение

В сообщении описан подход к интеграции интеллектуального анализа данных в реляционные СУБД на основе автоматизированной генерации алгоритмов ИАД, реализованных в виде хранимых процедур. Рассмотрена проблема автоматизированной генерации хранимых процедур. Приведено описание языка разметки Algorithm Explain Markup Language. Описан механизм выбора наиболее эффективного алгоритма ИАД из нескольких реализаций. Описана модульная структура и интерфейс программной системы, реализующей данный подход.

Перспективным направлением продолжения исследований по описанной тематике является адаптация предложенного подхода к использованию в контексте параллельных СУБД [11–13], в частности, для параллельной СУБД PargreSQL [14].

Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта № 12-07-00443-а.

Литература

1. Ordonez, C. Integrating K-Means Clustering with a Relational DBMS Using SQL // IEEE Transactions on Knowledge and Data Engineering. – 2006. – Vol. 18, No. 2. – P. 188–201.
2. Berthold, M.R. KNIME: The Konstanz Information Miner / M.R. Berthold, N. Cebron, F. Dill, et al. // Proceedings of the 31st Annual Conference of the Gessellschaft fur Klassifikation. – Springer Berlin Heidelberg, 2008. – P. 319–326.
3. Пан, К.С. Параллельный алгоритм решения задачи анализа рыночной корзины на процессорах Cell / К.С. Пан, М.Л. Цымблер // Вестник ЮУрГУ. Серия «Математическое моделирование и программирование». – 2010. – № 16(192). – Вып. 5. – С. 48–57.
4. Chung, S.M. Mining Association Rules from Relations on a Parallel NCR Teradata Database System / S.M. Chung, M. Mangamuri // Proc. of Int. Scientific Conference on Inf. Technology: Coding and Computing, 2004 (ITCC 2004). – Vol. 1. – P. 465–470.
5. Wang, F. SQL Implementation of a ScanOnce Algorithm for Large Database Mining / F. Wang, J. Gordon, N. Helian // Proceedings of the 5th Workshop on Engineering Federated Information Systems (EFIS 2003). – IOS Press, 2003. – P. 43–45.
6. Миниахметов, Р.М. Интеграция алгоритма кластеризации Fuzzy c-Means в PostgreSQL / Р.М. Миниахметов, М.Л. Цымблер // Вычислительные методы и программирование: Новые вычислительные технологии (Электронный научный журнал). – 2012. – Т. 13. – С. 46–52.
7. Miniakhmetov, R.M. Integrating Fuzzy c-Means Clustering with PostgreSQL // Труды Института системного программирования РАН. – 2011. – Т. 21. – С. 263–276.

8. Ordonez, C. A Data Mining System Based on SQL Queries and UDFs for Relational Databases // Proceedings of the 20th ACM International Conference on Information and Knowledge Management. – ACM, 2011. – P. 2521–2524.
9. Agrawal, R. Fast Algorithms for Mining Association Rules in Large Databases / R. Agrawal, R. Srikant // Proceedings of the 20th International Conference on Very Large Data Bases. – Morgan Kaufmann, 1994. – P. 487–499.
10. Ioannidis, Y.E. Query Optimization // ACM Computing Surveys. – 1996. – Vol. 28, No. 1. – P. 121–123.
11. Лепихов, А.В. Обработка запросов в СУБД для кластерных систем / А.В. Лепихов, Л.Б. Соколинский // Программирование. – 2010. – № 4. – С. 25–39.
12. Соколинский, Л.Б. Организация параллельного выполнения запросов в многопроцессорной машине баз данных с иерархической архитектурой // Программирование. – 2001. – № 6. – С. 13–29.
13. Соколинский, Л.Б. Обзор архитектур параллельных систем баз данных // Программирование. – 2004. – № 6. – С. 49–63.
14. Пан, К.С. Разработка параллельной СУБД на основе последовательной СУБД PostgreSQL с открытым исходным кодом / К.С. Пан, М.Л. Цымблер // Вестник ЮУрГУ. Серия «Мат. моделирование и программирование». – 2012. – № 18(277). – Вып. 12. – С. 112–120.

Тимофей Валерьевич Речкалов, аспирант кафедры системного программирования, Южно-Уральский государственный университет (г. Челябинск), trechkalov@gmail.com.

AN APPROACH TO INTEGRATION OF DATA MINING WITH RELATIONAL DBMS BASED ON AUTOMATIC SQL CODE GENERATION

T.V. Rechkalov, South Ural State University (Chelyabinsk, Russian Federation)

The paper introduces an approach to integration of data mining algorithms with relational DBMS. Approach assumes using XML-based markup for data mining algorithms implemented in SQL. User specifies input tables and algorithm parameters. After that system generates persistent stored procedures for specific data mining algorithm. Proposed markup language is described with examples. If there are several algorithms for some data mining problem the most effective algorithm implementation in SQL is generated by means of DBMS command EXPLAIN which shows estimated execution cost. Module structure and API for proposed program system is described.

Keywords: data mining, relational databases, SQL.

References

1. Ordonez C. Integrating K-Means Clustering with a Relational DBMS Using SQL. IEEE Transactions on Knowledge and Data Engineering, 2006. Vol. 18, No. 2. P. 188–201.
2. Berthold M.R., Cebron N., Dill F. et al. KNIME: The Konstanz Information Miner. Proceedings of the 31st Annual Conference of the Gessellschaft fur Klassifikation, 2008. Springer Berlin Heidelberg: 2008. P. 319–326.

3. Pan K.S. Parallel'nyj algoritm reshenija zadachi analiza rynochnoj korziny na processorah Cell [Parallel Algorithm for Market-Basket Problem on Cell Processors]. Vestnik JuUrGU. Serija «Mat. modelirovanie i programmirovanie» [Bulletin of the SUSU. Series «Mathematical modeling and programming»]. 2010. № 16(192). Vol. 5. P. 48–57.
4. Chung S.M., Mangamuri M. Mining Association Rules from Relations on a Parallel NCR Teradata Database System. Proceedings of International Scientific Conference on Information Technology: Coding and Computing, 2004 (ITCC 2004). Vol. 1. P. 465–470.
5. Wang F., Gordon J., Helian N. SQL Implementation of a ScanOnce Algorithm for Large Database Mining. Proceedings of the 5th Workshop on Engineering Federated Information Systems (EFIS 2003). IOS Press, 2003. P. 43–45.
6. Miniakhmetov R.M., Tsymbler M.L. Integracija algoritma klasterizacii Fuzzy c-Means v PostgreSQL [Integration of Fuzzy c-Means Clustering Algorithm with PostgreSQL Database Management System]. Vychislitelnye metody i programmirovanie: Novye vychislitelnye tehnologii (Jelektronnyj nauchnyj zhurnal) [Numerical Methods and Programming. Advanced Computing (Electronic Scientific Journal)]. 2012. Vol. 13. P. 46–52.
7. Miniakhmetov R.M. Integrating Fuzzy c-Means Clustering with PostgreSQL. Trudy Instituta sistemnogo programmirovanija RAN [Proc. of Institute for System Programming Russian Academy of Sciences]. 2011. Vol. 21. P. 263–276.
8. Ordonez C. A Data Mining System Based on SQL Queries and UDFs for Relational Databases. Proceedings of the 20th ACM International Conference on Information and Knowledge Management. ACM, 2011. P. 2521–2524.
9. Agrawal R., Srikant R. Fast Algorithms for Mining Association Rules in Large Databases. Proceedings of the 20th International Conference on Very Large Data Bases. Morgan Kaufmann, 1994. P. 487–499.
10. Ioannidis Y.E. Query Optimization // ACM Computing Surveys, 1996. Vol. 28, No. 1. P. 121–123.
11. Lepikhov A.V., Sokolinsky L.B. Query Processing in a DBMS for Cluster Systems. Programming and Computer Software. 2010. Vol. 36. No. 4. P. 205–215.
12. Sokolinsky L.B. Organization of Parallel Query Processing in Multiprocessor Database Machines with Hierarchical Architecture. Programming and Computer Software. 2001. Vol. 27. № 6. P. 297–308.
13. Sokolinsky L.B. Survey of Architectures of Parallel Database Systems. Programming and Computer Software. 2004. Vol. 30. № 6. P. 337–346.
14. Pan K.S., Zymbler M.L. Razrabotka parallel'noj SUBD na osnove posledovatel'noj SUBD PostgreSQL s otkrytym ishodnym kodom [Development of a Parallel Database Management System on the Basis of Open-Source PostgreSQL DBMS]. Vestnik JuUrGU. Serija «Mat. modelirovanie i programmirovanie» [Bulletin of the SUSU. Series «Mathematical modeling and programming»]. 2012. № 18(277). Vol. 12. P. 112–120.

Поступила в редакцию 6 ноября 2012 г.

СВЕДЕНИЯ ОБ ИЗДАНИИ

Серия основана в 2012 году.

Свидетельство о регистрации ПИ ФС77-26455 выдано 13 декабря 2006 г. Федеральной службой по надзору за соблюдением законодательства в сфере массовых коммуникаций и охране культурного наследия.

ПРАВИЛА ДЛЯ АВТОРОВ

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

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, Южно-Уральский государственный университет, факультет Вычислительной математики и информатики, кафедра СП, ответственному секретарю, доценту Цымблеру Михаилу Леонидовичу.
3. Адрес электронной почты редакции: `vestnikvmi@gmail.com`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**
5. 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).