ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

ВЕСТНИК ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

BULLETIN OF THE SOUTH URAL STATE UNIVERSITY

СЕРИЯ

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА

2022, том 11, № 3

SERIES

COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING

2022, volume 11, no. 3

1943

ВЕСТНИК

ЮЖНО-УРАЛЬСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА

2022
Т. 11, № 3

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

СЕРИЯ

«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»

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

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

Тематика журнала:

- Вычислительная математика и численные методы
- Математическое программирование
- Распознавание образов
- Вычислительные методы линейной алгебры
- Решение обратных и некорректно поставленных задач
- Доказательные вычисления
- Численное решение дифференциальных и интегральных уравнений
- Исследование операций
- Теория игр
- Теория аппроксимации
- Информатика
- Искусственный интеллект и машинное обучение
- Системное программирование
- Перспективные многопроцессорные архитектуры
- Облачные вычисления
- Технология программирования
- Машинная графика
- Интернет-технологии
- Системы электронного обучения
- Технологии обработки баз данных и знаний
- Интеллектуальный анализ данных

BULLETIN
OF THE SOUTH URAL STATE UNIVERSITY
SERIES
"COMPUTATIONAL MATHEMATICS AND SOFTWARE ENGINEERING"

2022
Vol. 11, no. 3

ISSN 2305-9052 (Print)
ISSN 2410-7034 (Online)

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

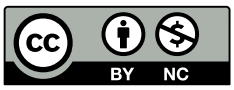
South Ural State University

The scope of the journal:

- Numerical analysis and methods
- Mathematical optimization
- Pattern recognition
- Numerical methods of linear algebra
- Reverse and ill-posed problems solution
- Computer-assisted proofs
- Numerical solutions of differential and integral equations
- Operations research
- Game theory
- Approximation theory
- Computer science
- Artificial intelligence and machine learning
- System software
- Advanced multiprocessor architectures
- Cloud computing
- Software engineering
- Computer graphics
- Internet technologies
- E-learning
- Database processing
- Data mining

Содержание

Contents



УДК 004.382.2, 004.4.42 **DOI: 10.14529/cmse220301**

ПРОГРАММНЫЕ СРЕДСТВА ВЫСОКОУРОВНЕВОГО СИНТЕЗА ДЛЯ МНОГОКРИСТАЛЬНЫХ РЕКОНФИГУРИРУЕМЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

© 2022 А.И. Дордопуло[1], И.И. Левин[1,2], В.А. Гудков[1,2], А.А. Гуленок[1]

[1]*ООО «НИЦ супер-ЭВМ и нейрокомпьютеров»*

(347922 Таганрог, пер. Итальянский, д. 106),

[2]*Южный федеральный университет (347928 Таганрог, пер. Некрасовский, д. 44)*

E-mail: dordopulo@superevm.ru, iilevin@sfedu.ru, gudkov@superevm.ru, gulenok@superevm.ru

Поступила в редакцию: 18.08.2022

В статье описывается оригинальный комплекс высокоуровневого синтеза, преобразующий последовательные программы в схемотехническую конфигурацию специализированных аппаратных средств для реконфигурируемых вычислительных систем. Из исходной последовательной программы строится абсолютно-параллельная форма — информационный граф. Далее, граф преобразуется в ресурсонезависимую параллельно-конвейерную форму — кадровую структуру, которую можно адаптировать к различному аппаратному ресурсу. Преобразование кадровой структуры в информационно-эквивалентную, но занимающую меньший аппаратный ресурс, структуру выполняется с помощью формализованных методов редукции производительности, что позволяет автоматически получить рациональное решение для заданной многокристальной реконфигурируемой вычислительной системы. В отличие от известных средств высокоуровневого синтеза результатом преобразования является не IP-ядро вычислительно-трудоемкого фрагмента, а автоматически синхронизированное решение прикладной задачи для всех кристаллов ПЛИС реконфигурируемой вычислительной системы. По сравнению с распараллеливающими компиляторами, число анализируемых вариантов синтеза рационального решения существенно меньше, что является отличительной особенностью описываемого комплекса. Применение программных средств высокоуровневого синтеза рассматривается на примере задачи решения системы линейных алгебраических уравнений методом Гаусса, содержащей информационно-взаимозависимые вычислительные фрагменты с существенно разной степенью параллелизма.

Ключевые слова: высокоуровневый синтез, трансляция программ, язык C, редукция производительности, реконфигурируемые вычислительные системы, программирование многопроцессорных вычислительных систем.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Дордопуло А.И., Левин И.И., Гудков В.А., Гуленок А.А. Программные средства высокоуровневого синтеза для многокристальных реконфигурируемых вычислительных систем // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 5–21. DOI: 10.14529/cmse220301.

Введение

Основной прагматичной целью высокопроизводительных вычислений является сокращение времени решения задачи. Существенное сокращение возможно путем повышения быстродействия элементной базы многопроцессорной вычислительной системы (МВС), применения специальных форм организации вычислений или максимального распараллеливания операций задачи [1]. Возможности микроэлектроники по минимизации размеров транзистора практически достигли предела, поэтому перспективы дальнейшего роста производительности с помощью повышения скорости работы ядер и увеличения их числа существенно ограничены. Распараллеливание вычислительных операций на десятки миллионов вычислительных узлов современных суперкомпьютеров, таких как Summit, Fugaku,

TaihuLight [2], приводит к обратному эффекту, если информационные зависимости в структуре решаемой задачи не учитываются: при увеличении числа узлов время решения задачи не сокращается, а увеличивается. Поэтому применение специальных форм организации вычислений с учетом информационных зависимостей между вычислительными фрагментами является наиболее перспективным направлением поиска способов сокращения времени решения задачи.

Учет информационных зависимостей — это одно из положений концепции структурно-процедурной организации вычислений [3] для реконфигурируемых вычислительных систем (РВС) с программируемыми логическими интегральными схемами (ПЛИС). РВС, содержащие множество объединенных коммутационной системой кристаллов ПЛИС [4], значительно превосходят МВС кластерной архитектуры по энергоэффективности и реальной производительности, но сильно уступают в удобстве программирования и отладки. Одним из многообещающих направлений упрощения программирования РВС является поиск новых решений в области средств трансляции последовательных программ в конфигурационные файлы ПЛИС с помощью программных средств высокоуровневого синтеза [5].

В настоящей работе рассматриваются методы преобразования входной программы на языке C в кадровую структуру и методы автоматического масштабирования полученных решений для доступного аппаратного ресурса заданной РВС с помощью механизмов редукции производительности. Во втором разделе рассматриваются известные средства высокоуровневого синтеза и особенности их работы, формулируются ключевые отличия описываемого комплекса от существующих. В третьем разделе приведены теоретические основы преобразований, используемых программными средствами высокоуровневого синтеза для автоматической адаптации прикладной задачи к доступному аппаратному ресурсу РВС. Четвертый раздел описывает основные этапы выполняемых программными средствами преобразований. В пятом разделе представлены результаты, полученные при трансляции задачи решения системы линейных алгебраических уравнений (СЛАУ) методом Гаусса. В заключении обобщаются предложенные принципы и методы.

1. Обзор существующих средств высокоуровневого синтеза

Для программирования ускорителей на ПЛИС, наряду с традиционными системами схемотехнического проектирования, такими как Xilinx Vivado или Intel Quartus Prime, все чаще используются средства высокоуровневого синтеза (High Level Synthesis, HLS). HLS-компиляторы — это трансляторы [5, 6] с некоторого языка программирования в конфигурационные файлы ПЛИС, преобразующие программы в схемотехническую конфигурацию специализированных аппаратных средств на языках HDL-группы. В зависимости от языка входной программы средства HLS [5] можно отнести к одной из двух категорий (рис. 1): трансляторы проблемно-ориентированных языков, т.е. адаптированных к определенной проблемной области версий языков программирования, и трансляторы языков общего назначения, т.е. диалектов языков с некоторыми особенностями и ограничениями. Из рис. 1 видно, что наибольшее число HLS-компиляторов относится к группе процедурных языков общего назначения на основе диалектов языка программирования C. В этой группе представлены как академические (DWARV [7], BAMBU [8] и LEGUP [9]), так и активно развивающиеся коммерческие (CatapultC, Vivado HLS [10], Vivado Vitis [11]) HLS-компиляторы, сравнительный анализ возможностей которых представлен в [5, 6].

Vivado HLS [10] — это САПР Xilinx, предназначенная для создания цифровых устройств с применением языков высокого уровня. Vivado HLS содержит ряд средств оптимизации, которые характерны как для компиляторов, так и для систем разработки цифровых схем [12]:

- конвейеризацию операций, которая планирует последовательность выполнения операций в пределах заданного числа тактов;
- анализ и оптимизацию разрядности для сокращения количества битов, передаваемых в информационных каналах;
- выделение внутренней памяти BRAM для быстрого доступа к памяти при низких аппаратных затратах;
- оптимизацию (конвейеризацию) циклов, которая использует параллелизм на уровне цикла для запуска очередной итерации цикла до завершения предшествующей с учетом информационной зависимости;
- пространственное распараллеливание для одновременной аппаратной реализации информационно-независимых команд;
- оптимизацию выполняемых операций, заменяющую выполняемую операцию информационно-эквивалентной, но требующей меньше затрат или ресурсов ПЛИС;
- прогнозирование и перемещение кода для его выноса из условных и управляющих конструкций;
- преобразования условий для планирования параллельных команд из непересекающихся путей выполнения.

Xilinx позиционирует Vivado HLS как инструмент разработки проектов, где важна скорость разработки, а не эффективность кода и/или сокращение аппаратных затрат при реализации в ПЛИС.

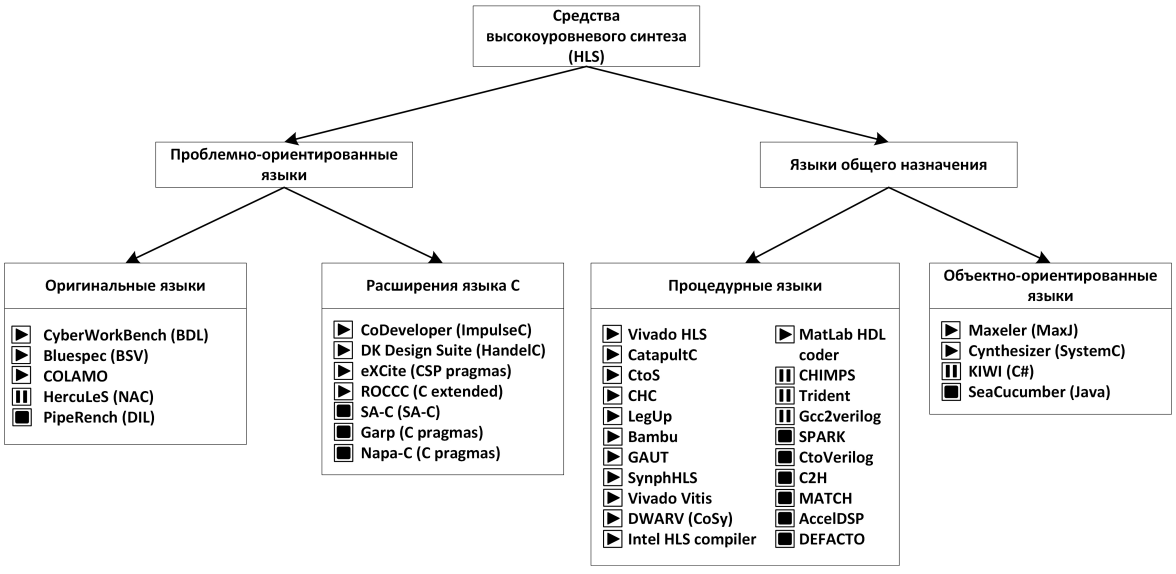


Рис. 1. Классификация средств высокоуровневого синтеза
(► — используется, ‖ — недоступен, ■ — завершен)

Xilinx Vitis [11] — это новая среда разработки, объединяющая графические инструменты, компиляторы, анализаторы и отладчики для ускорения фрагментов кода последовательных программ. Vitis содержит различные пакеты для решения отдельных задач, таких как Vitis Core для разработки ускорительных ядер широкого круга задач и Vitis AI для

ускорения работы систем искусственного интеллекта на базе нейронных сетей. Vitis ориентирован на работу со встроенными платформами Xilinx – ускорителями на базе ПЛИС для серверных и облачных приложений и/или ускорителями Alveo для встраиваемых устройств. Целевая платформа Vitis определяет базовую аппаратную и программную архитектуру и контекст приложения для платформ Xilinx, включая интерфейсы внешней памяти, пользовательские интерфейсы ввода/вывода и библиотеки времени исполнения.

Несмотря на обилие реализованных преобразований и методов оптимизации, применение Vivado HLS и Vitis не гарантирует автоматического роста производительности, т.к. далеко не каждая программа на языке C может быть эффективно реализована в ПЛИС [12]. Как и Vivado HLS/Vitis, подавляющее большинство приведенных в [5, 6] HLS-компиляторов, анализируют вычислительно трудоемкий фрагмент программы на языке C и преобразуют его в специализированный вычислитель (IP-ядро), синтезируемый на основе автоматной модели или процессорной парадигмы. Несмотря на существенный выигрыш полученного IP-ядра в скорости вычислений по сравнению с процессорной реализацией [5], масштабирование решения даже в пределах одного кристалла и организация потоков данных возлагаются на пользователя. Поэтому, несмотря на автоматизацию трансляции фрагмента последовательной программы, поиск рационального решения всей задачи для доступного аппаратного ресурса является целиком и полностью задачей программиста. При использовании нескольких ПЛИС, связанных пространственной коммутационной системой, или многокристальных РВС [3, 4] эта задача многократно усложняется и становится сродни масштабированию решений распараллеливающим компилятором. Распараллеливающий компилятор из последовательной программы восстанавливает параллелизм исходного алгоритма, выявляет участки и фрагменты, которые можно выполнить одновременно (например, итерации циклов), и добавляет инструкции для их параллельного исполнения на узлах МВС. Вычислительная сложность автоматического распараллеливания для МВС с распределенной памятью состоит в том, что для каждого варианта распараллеливания фрагментов последовательной программы компилятору нужно оценивать и ранжировать варианты размещения данных по процессорам с учетом характеристик коммуникационной сети, что многократно расширяет пространство перебора. Неоднократные попытки различных исследователей [13] создать стабильно развивающийся автоматизированный распараллеливающий компилятор для современных кластерных МВС с распределенной памятью показали [13], что для реальных, а не тестовых, задач это возможно только в крайне редких случаях из-за чрезвычайно большого числа вариантов и отсутствия удобного критерия для оценки эффективности синтезируемых версий параллельной программы.

В отличие от рассмотренных HLS-средств и распараллеливающих компиляторов авторами разработан оригинальный метод адаптации вычислений прикладной задачи к доступному вычислительному ресурсу многокристальных РВС. Согласно классификации (рис. 1), разработанный комплекс занимает промежуточное положение: исходной формой представления задачи является процедурный язык общего назначения C, который транслируется в программу на проблемно-ориентированном языке высокого уровня COLAMO [3], а далее — в конфигурационные файлы ПЛИС. При схожей функциональности от наиболее близких коммерческих аналогов, таких как Xilinx Vivado HLS и Vitis, комплекс отличается автоматическим преобразованием входной программы без указаний пользователя в виде директив #pragma или других способов ручной разметки кода, поддержкой многокристальных решений и автоматической синхронизацией информационных и управляющих сигналов.

2. Теоретические основы адаптации вычислений прикладной задачи

В отличие от рассмотренных HLS-компиляторов, последовательная программа в комплексе представляется в абсолютно параллельной форме в виде информационного графа [4] — ориентированного графа, вершины которого соответствуют операциям над данными, а дуги отражают информационную зависимость между ними. Каждая операция над элементами данных представляется операционной вершиной, поэтому общее число операционных вершин соответствует числу операций над данными задачи, а число входных вершин соответствует размерности всех обрабатываемых данных. Операционные вершины информационного графа задачи (ИГЗ) распределены по слоям и итерациям: слои, как и ярусы графа алгоритма [1], содержат информационно-независимые вершины, поэтому связи между вершинами в слое отсутствуют, а итерации описывают информационную зависимость вершин разных слоев между собой. ИГЗ является ациклическим, т.е. не содержит обратных связей. Операционные вершины в слоях и итерациях ИГЗ обычно представляют подграфы (например, соответствующие телу цикла в последовательных программах) из нескольких простых арифметических операций, связанными информационными зависимостями, среди которых могут быть условные или коммутационные операции. Информационный граф может быть достаточно просто построен из последовательной программы с помощью развертки циклов: на рис. 2 представлен фрагмент последовательной программы реализации прямого хода решения системы линейных алгебраических уравнений (СЛАУ) методом Гаусса на языке С.

```
for (i = 0; i < N; i++) {
  for (j = i+1 ; j < N ; j++) {
    d = (m[j][i]/m[i][i]);
   for (k = i; k < N+1 ; k++)
      m[j][k] = m[j][k] - (m[i][k]*d);
   }
}
```

Рис. 2. Фрагмент программной реализации прямого хода решения СЛАУ методом Гаусса на языке С

В слоях информационного графа решения СЛАУ методом Гаусса (рис. 3) расположены информационно-независимые подграфы g^i_{jk}, каждый из которых содержит 3 операционные вершины: деление, умножение и вычитание, что соответствует операциям в теле цикла по переменным **j** и **k** (рис. 2). Информационная зависимость между итерациями задается связями между подграфами: выход m^1 [1,1] подграфа $g^0_{1,1}$ нулевой итерации является входом подграфов $g^1_{2..N,1..N}$ первой, выход m^2 [3,2] подграфа $g^1_{3,2}$ является входом $g^2_{3..N,2..N}$ второй и т.д.). Согласно коду программы (рис. 2) каждый следующий слой ИГЗ содержит меньшее число подграфов g^i_{jk}.

Наиболее важным свойством ИГЗ является инвариантность его формы для различных способов описания прикладной задачи. ИГЗ — это абсолютно-параллельная форма представления вычислений, отражающая информационные зависимости между функционально-завершенными фрагментами программы, поэтому, с точностью до топологической перестановки подграфов в слое, ИГЗ будет неизменен для разных последовательных программ, описывающих одну и ту же прикладную задачу, при одинаковом базисе

используемых вычислительных операций. Так, при изменении кода программы (рис. 2), например, при разбиении внешнего цикла на два, ИГЗ не изменится, а для распараллеливающего компилятора измененная программа не будет тождественна исходной, поэтому при автоматическом распараллеливании этих семантически одинаковых программ будут получены разные варианты распараллеливания.

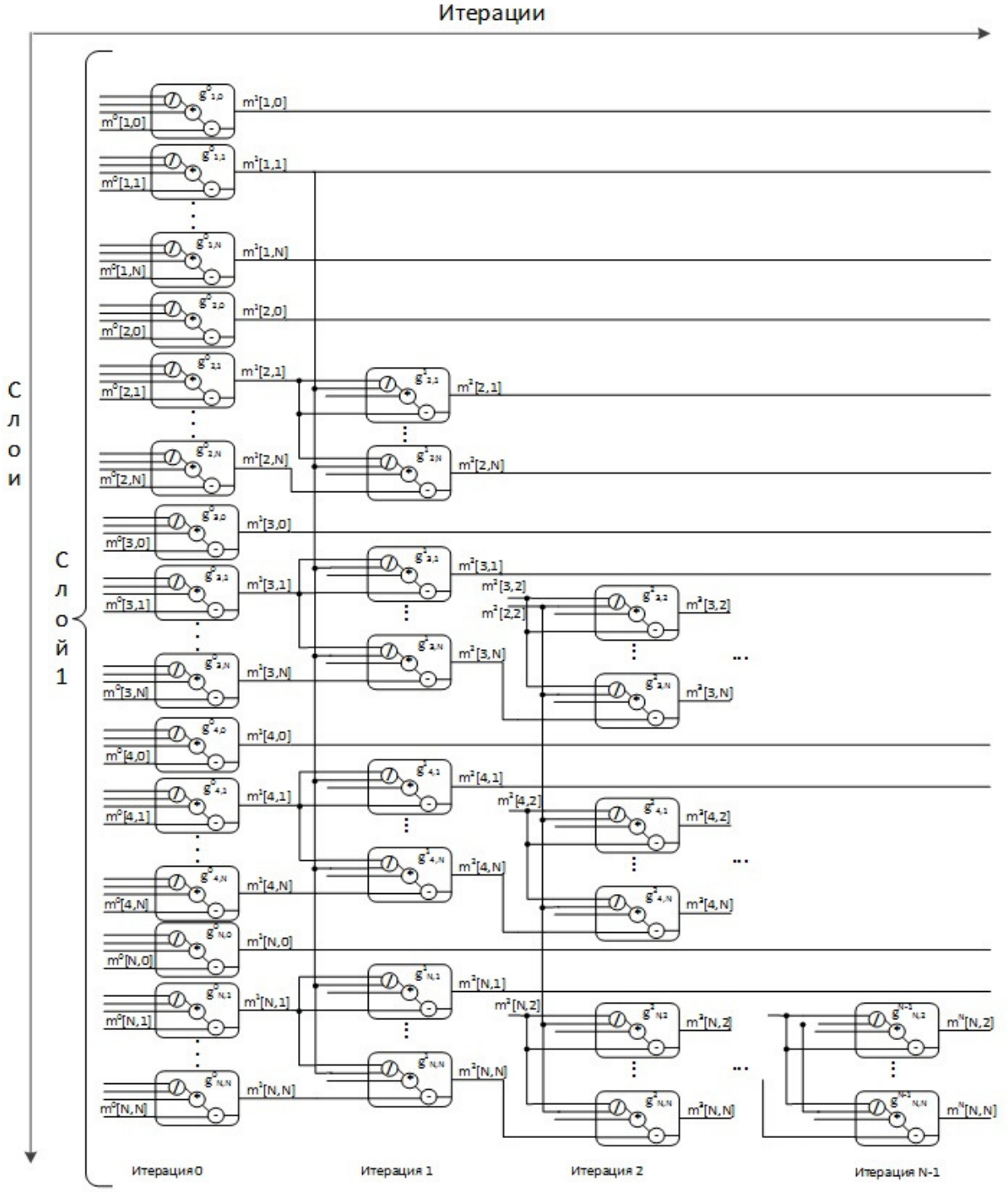


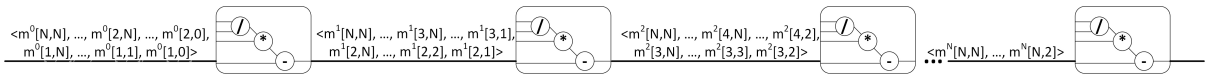
Рис. 3. Информационный граф прямого хода решения СЛАУ методом Гаусса

Инвариантность представления параллельных вычислений ИГЗ позволяет объединять топологически различные, но информационно-эквивалентные, варианты описания вычис-

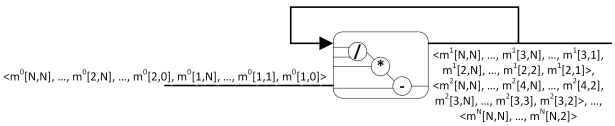
лений, что существенно сокращает число потенциальных анализируемых вариантов организации параллельных вычислений уже на уровне информационной модели. Аппаратная реализация ИГЗ в виде вычислительной структуры имеет минимальную латентность, наименьшее время решения и максимальную производительность, но требует множество вычислительных устройств и каналов памяти для одновременной подачи всех заданных параметрами задачи входных данных и выполнения всех операций, что обычно недостижимо для реальных вычислительных систем. Поэтому для реализации в вычислительной системе ИГЗ преобразуется в кадровую структуру [3], которая учитывает информационные зависимости между его фрагментами и основные параметры вычислительных устройств. Переход от информационного графа к кадровой структуре выполняется заменой операционных вершин графа на соответствующие операциям вычислительные устройства, а дуг — на связи коммутационной системы. Полученная в результате абсолютно параллельная кадровая структура (АПКС) отличается от ИГЗ учетом характеристик реализации на заданной вычислительной системе (частоты работы и быстродействия устройств, интервала обработки данных и др.). Например, ИГЗ прямого хода решения СЛАУ методом Гаусса (рис. 3) будет соответствовать абсолютно параллельной кадровой структуре при замене операций подграфов вычислительными устройствами при неизменной общей вычислительной структуре.

АПКС может быть преобразована в другие кадровые структуры, более экономичные по занимаемому аппаратному ресурсу и обеспечивающие информационную эквивалентность результатов вычислений. Если сократить ресурс, занимаемый АПКС (рис. 3), путем упорядочивания подграфов по слоям, получится приведенная на рис. 4-а структура, содержащая все итерации исходной АПКС с одним структурно-реализованным подграфом в каждом слое. При дальнейшем сокращении занимаемого аппаратного ресурса будет получена минимальная кадровая структура прямого хода решения СЛАУ методом Гаусса (рис. 4-б), которая содержит только базовый подграф, изоморфный остальным подграфам задачи, расположенным в слоях и итерациях. Каждая кадровая структура (рис. 3, 4) обладает различными характеристиками быстродействия и занимаемого ресурса, поэтому будем считать, что преобразование кадровых структур параметризуется аппаратным ресурсом, влияющим на время решения задачи.

Параметризуемое аппаратным ресурсом преобразование абсолютно параллельной кадровой структуры существенно отличается не только от методов распараллеливания, но и от применявшейся для РВС технологии индуктивных программ [4], также зависевшей от доступного ресурса.



а) кадровая структура, полученная методом распараллеливания по итерациям



б) минимальная кадровая структура

Рис. 4. Кадровые структуры разных вариантов реализации решения СЛАУ методом Гаусса

В зависимости от доступного вычислительного ресурса, информационно-эквивалентные преобразования кадровых структур можно представить движением в трехмерном пространстве, заданном осями основных характеристик кадровой структуры: «Число слоев», «Число итераций», «Команды», «Разрядность», «Время» и «Интервал» (рис. 5). Исходной точкой является абсолютно параллельная кадровая структура с координатами $(L_{max}, It_{max}, T_{min})$. Целью движении является достижение области доступного аппаратного ресурса, которая задана числом каналов распределенной памяти по слоям и аппаратным ресурсом по итерациям. На каждом отрезке движения, показанном на рис. 5 стрелками, сокращаются степень параллелизма, производительность и аппаратные затраты абсолютно параллельной кадровой структуры до информационно-эквивалентной структуры, которая находится в области доступного ресурса.

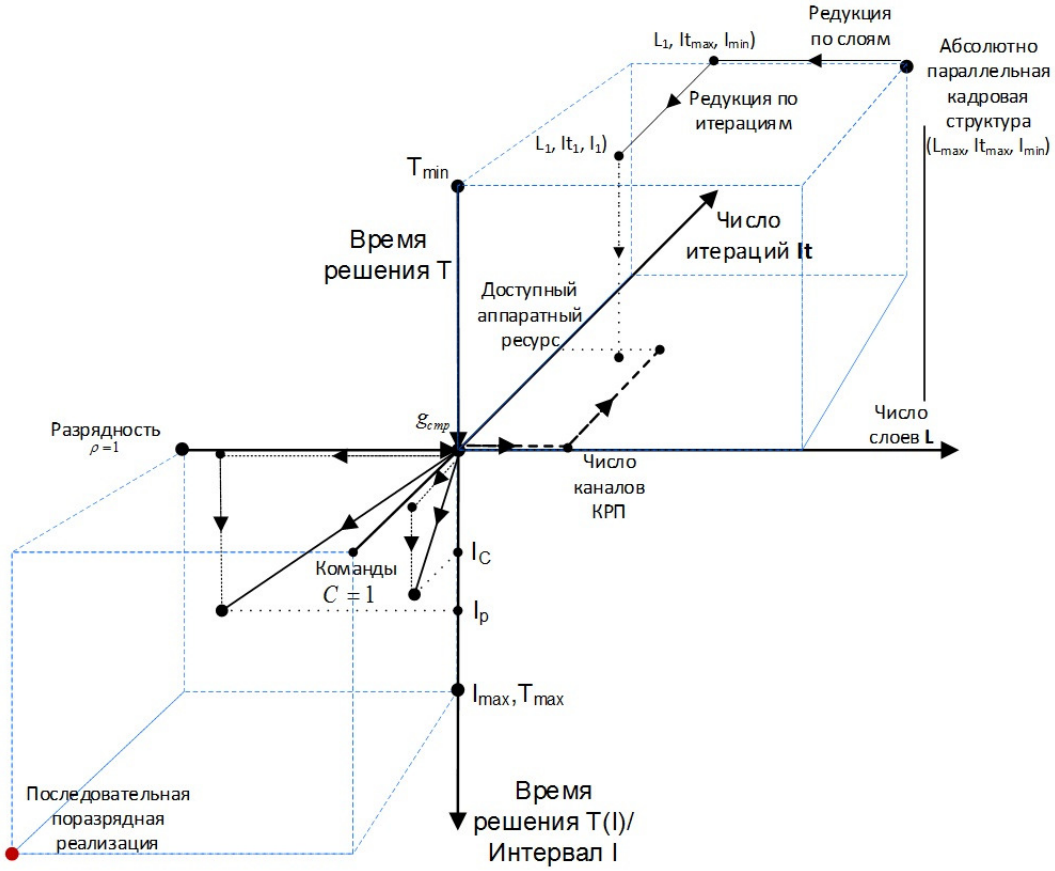


Рис. 5. Пространство реализации вычислений кадровых структур

Начальным направлением движения является сокращение числа каналов как типичного критического ресурса для большинства задач различных предметных областей, которое выполняется редукцией числа информационно-независимых слоевых подграфов ИГЗ, аппаратно реализуемых в кадровой структуре. Движение вдоль оси «Число слоев» завершается в некоторой точке с координатами (L_1, It_{max}, I_{min}), где L_1 соответствует числу доступных каналов либо предельному значению $L_1 = 1$. Дальнейшее движение происходит по оси «Число итераций» для сокращения числа информационно-зависимых подграфов (итераций) и завершается в некоторой точке (L_1, It_1, I_1), которая соответствует аппаратному ресурсу для размещения итерационных ступеней базовых подграфов g. Значение It_1 соответствует доступному ресурсу либо предельному значению $It_1 = 1$, а изменение I_{min} на I_1

отражает возможный при этом преобразовании рост интервала. Если область доступного аппаратного ресурса достигнута, кадровую структуру можно реализовать в архитектуре реконфигурируемой вычислительной системы. Для улучшения характеристик реализации и поиска наиболее рациональной кадровой структуры можно использовать оптимизационные преобразования, минимизирующие интервал обработки данных движением по оси «Интервал I».

Если даже при минимальных значениях слоев и итераций аппаратного ресурса вычислительной системы недостаточно для структурной реализации минимальной кадровой структуры из одного базового подграфа ИГЗ, выполняется переход в нижний октант пространства кадровых структур. Дальнейшее сокращение параллелизма кадровой структуры происходит с помощью редукции числа одновременно работающих устройств (команд) и разрядности одновременно обрабатываемых данных.

Переход от параллельного исполнения команд к последовательному — это привычный способ сокращения параллелизма при реализации вычислений на процессоре. Предельным значением параллелизма и ресурса кадровой структуры в нижнем октанте является последовательная поразрядная реализация вычислений на одном триггере. Поскольку ресурс реальных ПЛИС и РВС на их основе существенно больше минимального, движение гарантированно остановится в некоторой точке пространства, соответствующей кадровой структуре, реализуемой на доступном аппаратном ресурсе. После достижения области доступного аппаратного ресурса в нижнем октанте, возможно улучшить реализацию кадровой структуры с помощью оптимизационных преобразований — минимизировать интервал обработки данных движением по оси «Интервал I».

При движении в пространстве возможных реализаций, для текущей кадровой структуры на каждом шаге изменяется одна из ее характеристик: число слоев L_i^{it} или число итераций It_i^{it} или число команд (устройств) Q_i^{it} или разрядность ρ_i^{it} или интервал I_i^{it} обработки данных. Значения всех характеристик кадровой структуры целочисленные и представимы в виде произведения простых сомножителей, чтобы сократить производительность и аппаратные затраты наиболее рациональным способом. Минимизацию интервала обработки данных с помощью методов оптимизации будем считать отдельным этапом изменения характеристик кадровой структуры при движении в пространстве возможных реализаций.

Общее число редукционных преобразований, выполняемых для адаптации вычислений прикладной задачи к доступному вычислительному ресурсу РВС можно оценить сверху по сумме количества различных характеристик кадровой структуры, редуцируемых при движении в пространстве возможных реализаций: число слоев, числу итераций, числу команд, разрядности и интервалу обработки данных. Общее число этапов масштабирования для синтеза рациональной кадровой структуры определяется суммой числа этапов для достижения доступного ресурса РВС и числа оптимизационных преобразований. Согласно учитываемым характеритикам, общее число выполняемых преобразований не превысит шести, разумеется, при рациональном и корректном определении коэффициента редукции и нового значения сокращаемой характеристики кадровой структуры. Полученная оценка значительно меньше числа вариантов параллельной программы, анализируемых распараллеливающим компилятором, и инвариантна для разных версий исходной программы и ИГЗ, так и для разных прикладных задач за счет неизменного числа характеристик кадровой структуры. Это позволяет существенно сократить время портации АПКС при изменении архитектуры целевой вычислительной системы.

3. Компоненты комплекса средств высокоуровневого синтеза

Описанные принципы преобразования кадровой структуры прикладной задачи реализованы в комплексе средств высокоуровневого синтеза [14] для многокристальных реконфигурируемых вычислительных систем. Входная последовательная программа на языке С в стандарте ISO/IEC 9899:1999 преобразуется компонентами комплекса в ресурсонезависимую форму и масштабируется для доступного аппаратного ресурса РВС, а результатом трансляции является программа на языке высокого уровня COLAMO. Сокращение слоев, итераций, числа устройств, разрядности и интервала обработки данных при движении в пространстве возможных реализаций кадровых структур выполняется методами редукции производительности и аппаратных затрат, описанными в [15]. Все преобразования кадровой структуры выполняются следующими компонентами комплекса (рис. 6):

- транслятором «Ангел», преобразующим входную программу на языке С в информационный граф и абсолютно-параллельную кадровую структуру;
- процессором «Русалка», преобразующим АПКС в ресурсонезависимую параллельно-конвейерную форму, параметризуемую аппаратным ресурсом;
- выделение внутренней памяти BRAM для быстрого доступа к памяти при низких аппаратных затратах;
- процессором «Прокруст», выполняющим с помощью редукции производительности и аппаратных затрат движение в верхнем октанте и расчет параметров кадровой структуры для рациональной ее реализации в архитектуре заданной РВС;
- процессором «Щелкунчик», выполняющим редукцию производительности в нижнем октанте при нехватке аппаратного ресурса для структурной реализации базового подграфа ИГЗ.

Трансляция программы на языке COLAMO в конфигурационные файлы ПЛИС многокристальных РВС осуществляется разработанными ранее транслятором языка программирования COLAMO [3] и синтезатором многокристальных решений Fire!Constructor [3]. Загрузочные конфигурационные файлы (*.bit) синтезируются для каждого кристалла синтезатором системы автоматизированного проектирования Xilinx Vivado.

Методы преобразования последовательной программы с произвольным обращением к памяти в информационный граф задачи и абсолютно-параллельную кадровую структуру, выполняемое транслятором «Ангел», подробно рассмотрены в [16]. Анализ структуры АПКС, выделение подзадач, определение числа слоев и итераций для каждого фрагмента, анализ информационных зависимостей в структуре каждой подзадачи и между ними, расщепление скалярных переменных и растягивание массивов по итерациям для устранения нарушений правил однократного присваивания и единственной подстановки рассмотрены в [17]. Преобразование АПКС в масштабируемую параллельно-конвейерную форму выполняется процессором «Русалка» [17], а расчет параметров параллелизма задачи для доступного аппаратного ресурса — процессором «Прокруст». Если задача содержит несколько вычислительных фрагментов или подзадач с разной степенью параллелизма, необходимо сократить производительность всех подзадач в одинаковое число раз, заданное коэффициентом редукции производительности. При этом необходимо найти сбалансированное по интенсивности потоков данных решение для фрагментов задачи с разной степенью параллелизма, учитывая их информационные взаимозависимости и рациональную реализацию этого решения на доступном аппаратном ресурсе. Реализованная в процессоре «Прокруст» методика расчета параметров и преобразования кадровой структуры применима в том числе для

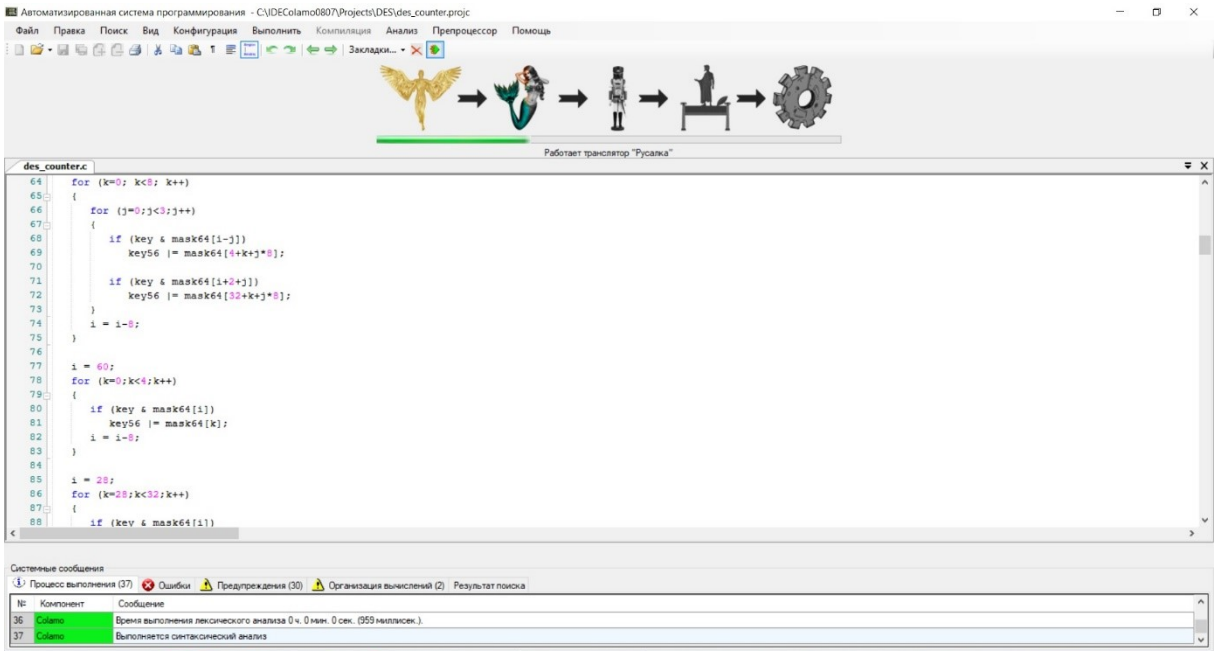


Рис. 6. Преобразование кадровой структуры компонентами комплекса средств высокоуровневого синтеза

задач, содержащих несколько вычислительных фрагментов или подзадач с разной степенью параллелизма. Для таких задач считается, что наиболее трудоемкий фрагмент вносит наибольший вклад в общее время решения задачи, поэтому именно этот фрагмент должен быть реализован наиболее эффективно, в наилучшем случае — в виде (мульти)конвейерной структуры с минимальным интервалом обработки данных. Для этого используется предложенное в [16] деление фрагментов задачи по величине занимаемого в АПКС аппаратного ресурса на «флагманские» фрагменты, занимающие наибольший ресурс, и занимающие существенно (как минимум, на один десятичный порядок) меньший ресурс «катера». Решение СЛАУ методом Гаусса, в котором можно выделить прямой и обратный ход, является одним из наиболее наглядных примеров такого разделения подзадач. Прямой ход по числу итераций циклов является «флагманом» с трудоемкостью $O(N^3)$ для исходной матрицы размерности N, а обратный ход представляет собой «катер» с трудоемкостью не более $O(N^2)$. Алгоритмы работы процессора «Прокруст» обеспечивают сбалансированное преобразование таких задач, содержащих фрагменты с разной степенью параллелизма.

4. Результаты экспериментальных исследований

С помощью комплекса средств высокоуровневого синтеза для многокристальных РВС реализован ряд прикладных программ линейной алгебры: решение систем линейных алгебраических уравнений методом Гаусса, методом Якоби для 3-диагональных матриц, методом Гаусса—Зейделя и разложением на верхнюю треугольную и нижнюю треугольную матрицы (LU-разложение). В табл. 1 представлены результаты экспериментальной проверки времени трансляции задачи решения СЛАУ методом Гаусса комплексом средств высокоуровневого синтеза. Размер обрабатываемых матриц составлял 8000×8001 элементов. Задача, содержащая прямой и обратный ход алгоритма, была реализована на трех РВС: «Тайгета» [4], «Терциус» и «Терциус-2». Для каждой из аппаратных платформ измерялись два показа-

теля: время портации (преобразования) кадровой структуры к архитектуре РВС и время синтеза решения. Время портации задачи на конфигурацию РВС прикладными программистами было принято равным одному 8-часовому рабочему дню, выраженному в секундах. Время синтеза решения комплексом и прикладными программистами определялось как сумма времени портации и времени синтеза загрузочного конфигурационного файла ПЛИС, которое зависит от логической емкости ПЛИС и уровня заполнения кристалла. Заполнение кристалла рассматривалось на уровне 90%, а время синтеза загрузочного конфигурационного файла составляет 6–8 часов для РВС «Тайгета», 8 часов для РВС «Терциус» и «Терциус-2».

Таблица 1. Результаты экспериментальной проверки времени портации задачи решения СЛАУ методом Гаусса

Аппаратные платформы	РВС «Тайгета»	РВС «Терциус»	РВС «Терциус-2»
Время портации на конфигурацию РВС комплексом, с	2 111.3	2 748.2	2 751
Время портации на конфигурацию РВС прикладными программистами, с	57 600	57 600	57 600
Время синтеза решения комплексом, с	23 711.3	31 548.2	31 548.2
Время синтеза решения прикладными программистами, с	79 200	86 400	86 400
Коэффициент сокращения временных затрат по времени портации	27.3	21	21
Коэффициент сокращения временных затрат по времени синтеза	3.3	2.7	2.7

Коэффициент сокращения временных затрат (по времени портации и времени синтеза) определялся как отношение соответствующей временной характеристики, полученной прикладными программистами, ко времени работы средств высокоуровневого синтеза. Достигнутый комплексом уровень реальной производительности определялся как отношение числа подграфов в решении, полученном разработанными средствами, к числу подграфов, полученных прикладными программистами. Для каждого полученного решения проверялась его работоспособность и корректность получаемых результатов запуском на соответствующей аппаратной платформе, а его реальная производительность сопоставлялась с результатами, полученными при портации ресурсонезависимой параллельной программы прикладными программистами для определения эффективности и достижения заданного уровня реальной производительности, представленных в табл. 2.

Прямой ход, являющийся «флагманом», с помощью распараллеливания по итерациям реализован структурно в виде итерационных ступеней обработки, связанных информационной зависимостью. Удельная производительность полученного средствами высокоуровневого синтеза варианта задачи решения СЛАУ методом Гаусса составила 0.83–0.94 от созданной прикладными программистами на языке COLAMO программы. Время трансляции

Таблица 2. Результаты экспериментальной проверки уровня производительности при портации задачи решения СЛАУ методом Гаусса

Аппаратные платформы	РВС «Тайгета»	РВС «Терциус»	РВС «Терциус-2»
Число подграфов в решении, полученном комплексом	784	1 178	1 178
Число подграфов в решении, полученном прикладными программистами	830	1 286	1 286
Уровень реальной производительности, достигнутый комплексом	0.94	0.91	0.91
Заданный уровень реальной производительности,	0.7		

задачи указанной размерности всеми компонентами комплекса составило 22 мин. 45 сек. на персональном компьютере с процессором Intel i5 7300 и 8 Гб оперативной памяти.

При трансляции этой же задачи в Vivado HLS полученное решение содержит 1 итерационную ступень, с помощью ручной разметки кода директивами #pragma удалось получить решение для 2 итерационных ступеней алгоритма прямого хода. Сравнение результатов времени портации и числа подграфов в решении, полученных комплексом, прикладными программистами и Vivado HLS позволяет сделать вывод о применимости предложенных теоретических основ для адаптации задачи решения СЛАУ методом Гаусса к различным РВС и более высокой эффективности для этой задачи по сравнению с компилятором Vivado HLS.

Заключение

Применение методов редукции производительности и аппаратных затрат для адаптации параллельных вычислений в новой версии комплекса позволяет значительно сократить число анализируемых вариантов при синтезе вычислительной структуры для РВС. В отличие от известных HLS-компиляторов, входная программа на языке С преобразуется комплексом автоматически, без ручной разметки кода или иных указаний пользователя. Комплекс поддерживает синтез многокристальных решений с автоматической синхронизацией информационных и управляющих сигналов. Проведено сравнение результатов трансляции задачи решения СЛАУ методом Гаусса описываемым комплексом и прикладными программистами, подтвердившее достижение заданных требований — автоматическое получение конфигурационных файлов ПЛИС для рационального многокристалльного решения (с эффективностью не ниже 50% от результатов, полученных инженерами-схемотехниками) за существенно меньшее (по сравнению с распараллеливающими компиляторами) число преобразований. Дальнейшее направление исследований — расширение полученных результатов на другие задачи математической физики и линейной алгебры позволит существенно повысить скорость разработки решения прикладной задачи и упростить портацию для многокристальных РВС различных архитектур и конфигураций.

Литература

1. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. БХВ-Петербург, 2002. 608 с.
2. Антонов А.С., Афанасьев И.В., Воеводин Вл.В. Высокопроизводительные вычислительные платформы: текущий статус и тенденции развития // Вычислительные методы и программирование. 2021. Т. 22, № 2. С. 135–177. DOI: 10.26089/NumMet.v22r210.
3. Гузик В.Ф., Каляев И.А., Левин И.И. Реконфигурируемые вычислительные системы. Таганрог: Изд-во ЮФУ, 2016. 472 с.
4. Levin I., Dordopulo A., Fedorov A., Kalyaev I. Reconfigurable computer systems: from the first FPGAs towards liquid cooling systems // Supercomputing Frontiers and Innovations. 2016. Vol. 3, no. 1. P. 22–40. DOI: 10.14529/jsfi160102.
5. Nane R., Sima V.-M., Pilato C., *et al.* A Survey and Evaluation of FPGA High-Level Synthesis Tools // IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2016. Vol. 35, no. 10. P. 1591–1604. DOI: 10.1109/TCAD.2015.2513673.
6. Numan M.W., Phillips B.J., Puddy G.S., Falkner K. Towards Automatic High-Level Code Deployment on Reconfigurable Platforms: A Survey of High-Level Synthesis Tools and Toolchains // IEEE Access. 2020. Vol. 8. P. 174692–174722. DOI: 10.1109/ACCESS.2020.3024098.
7. Nane R., Sima V.-M., Olivier B., *et. al.* DWARV 2.0: A CoSy-based C-to-VHDL Hardware Compile // 22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, August 29-31, 2012. IEEE, 2012. P. 619–622. DOI: 10.1109/FPL.2012.6339221.
8. Pilato C., Ferrandi F. Bambu: A Modular Framework for the High Level Synthesis of Memory-intensive Applications // 23rd International Conference on Field programmable Logic and Applications, FPL 2013, Porto, Portugal, September 2-4, 2013. IEEE, 2013. P. 1–4. DOI: 10.1109/FPL.2013.6645550.
9. Canis A., Choi J., Aldham M., *et. al.* LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems // Proceedings of the ACM/SIGDA 19th International Symposium on Field Programmable Gate Arrays, FPGA 2011, Monterey, California, USA, February 27 - March 1, 2011. ACM, 2011. P. 33–36. DOI: 10.1145/1950413.1950423.
10. Make Slow Software Run Fast with Vivado HLS. URL: `https://www.xilinx.com/publications/xcellonline/run-fast-with-Vivado-HLS.pdf` (дата обращения: 10.03.2021).
11. Vitis Unified Software Platform Documentation. Application Acceleration Development. URL: `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug1393-vitis-application-acceleration.pdf` (дата обращения: 10.03.2021).
12. Тарасов И. Проектирование для ПЛИС Xilinx с применением языков высокого уровня в среде Vivado HLS // Компоненты и технологии. 2013. № 12(149). С. 40–48.
13. Kolganov A.S. An experience of applying the parallelization regions for the step-by-step parallelization of software packages using the SAPFOR system // Numerical methods and programming. 2020. Vol. 21, no. 66. P. 388–404. DOI: 10.26089/NumMet.v21r432.
14. Levin I., Dordopulo A., Gudkov V., *et al.* Software Development Tools for FPGA-Based Reconfigurable Systems Programming // Russian Supercomputing Days. Vol. 1129 / ed. by

Vl. Voevodin, S. Sobolev. Cham: Springer, 2019. P. 625–640. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_51.

15. Dordopulo A.I., Levin I.I. Performance Reduction For Automatic Development of Parallel Applications For Reconfigurable Computer Systems // Supercomputing Frontiers and Innovations. 2020. Vol. 7, no. 2. P. 4–23. DOI: 10.14529/jsfi200201.

16. Левин И.И., Дордопуло А.И., Гудков В.А. и др. Комплекс средств программирования реконфигурируемых вычислительных систем на основе ПЛИС // Параллельные вычислительные технологии (ПаВТ'2020): Короткие статьи и описания плакатов, Пермь, 31 марта - 2 апреля, 2020. Челябинск: Издательский центр ЮУрГУ, 2020. С. 163–173. DOI: 10.14529/cmse150202.

17. Левин И.И., Дордопуло А.И., Гудков В.А. и др. Средства программирования реконфигурируемых и гибридных вычислительных систем на основе ПЛИС // Параллельные вычислительные технологии (ПаВТ'2019): Короткие статьи и описания плакатов XIII Международной научной конференции, Калининград, 2-4 апреля, 2019. Челябинск: Издательский центр ЮУрГУ, 2019. С. 299–312.

Дордопуло Алексей Игоревич, к.т.н., начальник, отдел математического и алгоритмического обеспечения, ООО «НИЦ супер-ЭВМ и нейрокомпьютеров» (Таганрог, Российская Федерация)

Левин Илья Израилевич, д.т.н., профессор, кафедра интеллектуальных и многопроцессорных систем, Южный федеральный университет, ООО «НИЦ супер-ЭВМ и нейрокомпьютеров» (Таганрог, Российская Федерация)

Гудков Вячеслав Александрович, к.т.н., доцент, кафедра интеллектуальных и многопроцессорных систем, Южный федеральный университет, ООО «НИЦ супер-ЭВМ и нейрокомпьютеров» (Таганрог, Российская Федерация)

Гуленок Андрей Александрович, к.т.н., старший научный сотрудник, отдел математического и алгоритмического обеспечения, ООО «НИЦ супер-ЭВМ и нейрокомпьютеров» (Таганрог, Российская Федерация)

DOI: 10.14529/cmse220301

HIGH-LEVEL SYNTHESIS SOFTWARE FOR MULTI-CHIP RECONFIGURABLE COMPUTING SYSTEMS

© 2022 A.I. Dordopulo[1], I.I. Levin[1,2], V.A. Gudkov[1,2], A.A. Gulenok[2]

[1] *Supercomputers and Neurocomputers Research Center (Italyansky lane 106, Taganrog, 347922 Russia),*

[2] *Southern Federal University (Nekrasovsky lane 44, Taganrog, 347928 Russia)*

E-mail: dordopulo@superevm.ru, iilevin@sfedu.ru, gudkov@superevm.ru, gulenok@superevm.ru

Received: 18.08.2022

The article describes an original complex of high-level synthesis that converts sequential programs into a circuit configuration of specialized hardware for reconfigurable computing systems. An absolutely parallel form, an information graph, is constructed from the original sequential program. Further, the graph is transformed into a resource-independent parallel–pipeline form — a personnel structure that can be adapted to various hardware resources. The transformation of the personnel structure into an information-equivalent structure, but occupying a smaller hardware resource, is performed using formalized methods of performance reduction, which allows you to automatically obtain a rational solution for a given multi-chip reconfigurable computing system. Unlike the known means of high-level synthesis, the result of the transformation is not the IP core of a computationally time-consuming fragment, but an automatically synchronized solution of an applied problem for all FPGA crystals of a reconfigurable computing system. Compared with parallelizing compilers, the number of analyzed variants of the synthesis of a rational solution is significantly less, which is a distinctive feature of the described complex. The application of high-level synthesis software is considered by the example of the problem of solving a system of linear algebraic equations by the Gauss method containing information-interdependent computational fragments with significantly different degrees of parallelism.

Keywords: high-level synthesis, program translation, C language, performance reduction, reconfigurable computing systems, programming of multiprocessor computing systems.

FOR CITATION

Dordopulo A.I., Levin I.I., Gudkov V.A., Gulenok A.A. High-level Synthesis Software for Multi-chip Reconfigurable Computing Systems. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 3. P. 5–21. (in Russian) DOI: 10.14529/cmse220301.

References

1. Voevodin V.V., Voevodin Vl.V. Parallel computing. BHV-Petersburg, 2002. 608 p. (in Russian).

2. Antonov A.S., Afanasyev I.V., Voevodin Vl.V. High-performance computing platforms: current status and development trends. Num. Meth. Prog. 2021. Vol. 22, no. 2. P. 135–177. DOI: 10.26089/NumMet.v22r210 (in Russian).

3. Guzik V.F., Kalyaev I.A., Levin I.I. Reconfigurable computer systems. Taganrog: SFEDU Publishing, 2016. 472 p. (in Russian).

4. Levin I., Dordopulo A., Fedorov A., Kalyaev I. Reconfigurable computer systems: from the first FPGAs towards liquid cooling systems. Supercomputing Frontiers and Innovations. 2016.

Vol. 3, no. 1. P. 22–40. DOI: 10.14529/jsfi160102.

5. Nane R., Sima V.-M., Pilato C., *et al.* A Survey and Evaluation of FPGA High-Level Synthesis Tools. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2016. Vol. 35, no. 10. P. 1591–1604. DOI: 10.1109/TCAD.2015.2513673.
6. Numan M.W., Phillips B.J., Puddy G.S., Falkner K. Towards Automatic High-Level Code Deployment on Reconfigurable Platforms: A Survey of High-Level Synthesis Tools and Toolchains. IEEE Access. 2020. Vol. 8. P. 174692–174722. DOI: 10.1109/ACCESS.2020.3024098.
7. Nane R., Sima V.-M., Olivier B., *et. al.* DWARV 2.0: A CoSy-based C-to-VHDL Hardware Compiler. 22nd International Conference on Field Programmable Logic and Applications (FPL), Oslo, Norway, August 29-31, 2012. IEEE, 2012. P. 619–622. DOI: 10.1109/FPL.2012.6339221.
8. Pilato C., Ferrandi F. Bambu: A Modular Framework for the High Level Synthesis of Memory-intensive Applications. 23rd International Conference on Field programmable Logic and Applications, FPL 2013, Porto, Portugal, September 2-4, 2013. IEEE, 2013. P. 1–4. DOI: 10.1109/FPL.2013.6645550.
9. Canis A., Choi J., Aldham M., *et. al.* LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems. Proceedings of the ACM/SIGDA 19th International Symposium on Field Programmable Gate Arrays, FPGA 2011, Monterey, California, USA, February 27 - March 1, 2011. ACM, 2011. P. 33–36. DOI: 10.1145/1950413.1950423.
10. Make Slow Software Run Fast with Vivado HLS. URL: `https://www.xilinx.com/publications/xcellonline/run-fast-with-Vivado-HLS.pdf` (accessed: 10.03.2021).
11. Vitis Unified Software Platform Documentation. Application Acceleration Development. URL: `https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug1393-vitis-application-acceleration.pdf` (accessed: 10.03.2021).
12. Tarasov I. Designing for Xilinx FPGAs using high-level languages in Vivado HLS environment. Components and Technologies. 2013. No. 12(149). P. 40–48 (in Russian).
13. Kolganov A.S. An experience of applying the parallelization regions for the step-by-step parallelization of software packages using the SAPFOR system. Numerical methods and programming. 2020. Vol. 21, no. 66. P. 388–404. DOI: 10.26089/NumMet.v21r432.
14. Levin I., Dordopulo A., Gudkov V., *et al.* Software Development Tools for FPGA-Based Reconfigurable Systems Programming. Russian Supercomputing Days. Vol. 1129 / ed. by Vl. Voevodin, S. Sobolev. Cham: Springer, 2019. P. 625–640. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_51.
15. Dordopulo A.I., Levin I.I. Performance Reduction For Automatic Development of Parallel Applications For Reconfigurable Computer Systems. Supercomputing Frontiers and Innovations. 2020. Vol. 7, no. 2. P. 4–23. DOI: 10.14529/jsfi200201.
16. Levin I.I., Dordopulo A.I., Gudkov V.A., *et al.* A set of programming tools for reconfigurable computing systems based on FPGA. Parallel Computational Technologies (PaVT'2020): Short articles and posters, Perm, March 31 - April 2, 2020. Chelyabinsk: SUSU Publishing Center, 2020. P. 163–173. DOI: 10.14529/cmse150202 (in Russian).
17. Levin I.I., Dordopulo A.I., Gudkov V.A., *et al.* Programming tools for reconfigurable and hybrid computing systems based on FPGA. Parallel Computational Technologies (PaVT'2019): Short articles and posters, Kaliningrad, April 2-4, 2019. Chelyabinsk: SUSU Publishing Center, 2019. P. 299–312 (in Russian).

УДК 19.175.3, 004.722.2, 004.436.4 DOI: 10.14529/cmse220302

МЕТОД ОПИСАНИЯ ТОПОЛОГИЧЕСКОЙ СТРУКТУРЫ ВЫЧИСЛИТЕЛЬНЫХ КЛАСТЕРОВ, ОСНОВАННЫЙ НА ОПЕРАЦИЯХ ПРОИЗВЕДЕНИЙ ПОДГРАФОВ*

© 2022 Э.Р. Хабирова[1], А.Н. Сальников[2,3]

[1] *ООО «Открытая Мобильная Платформа»*
(119270 Москва, пр. Вернадского, д. 41, 8 этаж),
[2] *Московский государственный университет имени М.В. Ломоносова*
(119991 Москва, ул. Ленинские горы, д. 1),
[3] *Федеральный исследовательский центр «Информатика и управление» РАН*
(119333 Москва, ул. Вавилова, д. 44, кор. 2)
E-mail: lineprinter0@gmail.com, salnikov@cs.msu.ru

Поступила в редакцию: 22.08.2022

Топологическая структура коммуникационных сетей суперкомпьютерных систем при увеличении размера и сложности суперкомпьютеров соответственно усложняется. Для ее описания существует множество методов, однако такие описания являются громоздкими, что усложняет манипулирование ими. В статье предложен подход к описанию коммуникационной среды суперкомпьютера, когда коммуникационная сеть описывается как конструктор, где элементами конструктора являются типовые топологические структуры, часто встречающиеся в различных вычислительных системах. С этой целью разработан язык описания топологической структуры, основанный на операции произведения подграфов. Язык идейно схож в своих принципах с языками NetML и OMNeT++. Отдельное внимание в работе уделяется исключениям в регулярности сетей реальных суперкомпьютеров; с целью добавления возможности описания данного факта в язык внесены специальные конструкции. Для поддержки работы с языком описания разработана библиотека на языке программирования Си и специальная оболочка над ней написанная на языке Python3, которая затем может использоваться для визуализации описываемых языком графов. Выразительная мощность языка была продемонстрирована на описании вычислительных кластеров: Tianhe-2A, AI Bridging Cloud Infrastructure и Ломоносов-2. Метод был проверен и сравнен с GraphViz DOT показано многократное сокращение необходимых объема записи для некоторых крупных систем из Top500.

Ключевые слова: вычислительный кластер, топология компьютерной сети, языки описания графов, произведения подграфов.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Хабирова Э.Р., Сальников А.Н. Метод описания топологической структуры вычислительных кластеров, основанный на операциях произведений подграфов // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 22–44. DOI: 10.14529/cmse220302.

Введение

Задачи, требующие огромных вычислительных ресурсов, выполняются на специальных вычислительных системах, позволяющих производить триллионы и квадриллионы операций с плавающей точкой в секунду — суперкомпьютерах. Современные суперкомпьютеры построены как вычислительный кластер серверов, объединенных сложной высокопроизводительной сетью. Размеры суперкомпьютерных систем увеличиваются: число узлов достигает десятков тысяч, а число вычислительных ядер — нескольких миллионов (с актуальным

*Статья рекомендована к публикации программным комитетом Международной научной конференции «Параллельные вычислительные технологии (ПаВТ) 2022».

рейтингом TOP 500 самых мощных суперкомпьютерных систем в мире можно ознакомиться в [1]). Коммуникационные сети, являющиеся «бутылочным горлышком», становятся одним из ключевых факторов в определении производительности.

Кроме этого, топологии суперкомпьютерных коммуникационных сетей постоянно развиваются. Существует множество распространенных сетевых топологий — начиная с более простых топологий, таких как «звезда», «n-мерная решетка», «n-мерный тор», «гиперкуб» (описанных в классическом учебнике [2]), заканчивая более сложными, но более эффективными (имеющими лучшие характеристики, такие как диаметр, связность, ширина бисекции, а также являющимися более экономически выгодными), такими как «утолщенное дерево» («fat tree», хорошо исследованная топология, описанная, например, в [3, 4]), «стрекоза» («dragonfly», впервые описанная в [5]), «плоская бабочка» («flattened butterfly», представленная в [6]), и другими. Более того, каждая топология имеет множество параметров, сильно влияющих на размеры и характеристики итоговой сети.

При таком разнообразии возможных топологий коммуникационных сетей и их возможных конфигураций, а также при увеличении размеров этих сетей усложняется задача выбора конкретной топологии при построении новых суперкомпьютерных систем, особенно принимая во внимание критичность этого выбора для производительности всей будущей системы в целом. В таком случае можно прибегнуть к моделированию суперкомпьютеров и их отдельных компонентов на системах меньшего размера.

Имитационное моделирование коммуникационных сетей суперкомпьютерных систем позволяет оценить характеристики коммуникационной сети и всей системы в целом. С его помощью можно оценить, подходит ли конкретная сеть именно для задач, которые планируется запускать на данном суперкомпьютере. Оно также является более экономически эффективным методом, чем, например, построение макета; анализ с помощью моделирования перед переходом к этапу анализа на макете позволяет сократить этот сложный и дорогостоящий этап. Кроме того, к моделированию коммуникационных сетей суперкомпьютеров прибегают при анализе производительности уже существующих систем, что может помочь использовать их более эффективно; яркий пример такого использования представлен в [7]. Подробнее про моделирование коммуникационных сетей суперкомпьютеров описано в [8, 9].

При увеличении размеров усложняется задача описания коммуникационной сети. При наличии в системе тысяч узлов, имеющих между собой еще большее число связей, установленных по неочевидным правилам, описания становятся нетривиальными, их размеры растут, а адресация конкретных узлов усложняется. При этом подробное описание сети необходимо не только при администрировании уже существующего суперкомпьютера, но и при моделировании, и в последнем случае требуется оперировать тем количеством описаний, сколько вариантов архитектуры сети рассматривается. Кроме того, в случае моделирования описания должны составляться и редактироваться человеком.

Топологии коммуникационных сетей суперкомпьютерных систем чаще всего имеют регулярную структуру. Даже в сетях, построенных для иных целей, можно наблюдать закономерности, что показано, например, в [10]. Возможно, этот факт можно использовать для их описания. Это могло бы как уменьшить размеры описаний, так и сделать написание их человеком более удобным, а также упростить адресацию отдельных узлов — в случае, если именование узлов можно вывести из структуры сети.

Целью является создать метод описания топологической структуры коммуникационных сетей суперкомпьютерных систем, подходящий для обмена описаниями, такой, чтобы:

- описания были компактней, чем перечисление всех узлов и всех ребер связей;
- описания использовали факт регулярной структуры сетей;
- с помощью описания можно было автоматически дать имя каждому узлу сети.

Имеет смысл рассмотреть существующие методы описания графов и методы описания сетей, используемые в системах моделирования. Рассмотрим их в контексте описания топологической структуры суперкомпьютеров.

Статья организована следующим образом: в разделе 2 представлены возможные подходы к описанию графов, в том числе некоторые популярные языки описаний; в разделе 3 подходы, применяемые именно к описанию сетей; в разделе 4 обсуждаются операции произведения подграфов и их применение к описанию сетей, так же приводится описание разработанного языка; в разделе 5 приводятся примеры базовых описаний и описаний некоторых популярных топологий вычислительных сетей; в разделе 6 приводятся результаты сравнения объемов описаний; в разделе 7 приводится описание функций созданых библиотек, для работы с описаниями графов; в разделе 8 некоторые обобщения.

1. Методы описания графов

Множество методов описания графов довольно обширно. Статья [11] описывает 76 файловых форматов, созданных для хранения и обмена графами. Авторы предлагают несколько классификаций графовых форматов. По одной из классификаций, все описанные файловые форматы разделяются на группы в соответствии с принципом представления графа. В соответствии с изложенным в статье доступные форматы представления графов можно разделить на:

- перечисляющие представления:
  - представляющие граф в виде списка всех ребер;
  - представляющие граф в виде матрицы смежности;
  - представляющие граф в виде разреженной матрицы;
  - представляющие граф в виде списка смежности;
  - представляющие граф в виде списка путей;
- процедурные, определяющие граф не перечислением вершин и ребер, а с помощью набора процедур. Такие форматы зачастую по сути используют многие библиотеки по работе с графами, например, Boost Graph Library [12].
- конструктивные, определяющие граф через операции на меньших графах.

В соответствии со сформулированными ранее целями работы оказывается, что первые пять вариантов (т.е. методы, основанные на перечислении; не являющиеся процедурными или конструктивными) нас не интересуют, поскольку подобные методы недостаточно компактны.

Процедурные методы описания графов в подавляющем большинстве случаев используют существующие языки программирования. По этой причине такие методы плохо подходят для обмена описаниями структур, поскольку описание, написанное, возможно, на другом языке программирования, разбирать достаточно сложно. Следовательно, они не соответствуют нашей цели.

Конструктивные методы лучше всего подходят для описания коммуникационных сетей суперкомпьютерных систем, поскольку такие сети зачастую имеют регулярную иерархическую или рекурсивную структуру.

Широко используемые языки **GraphViz DOT** [13], **TGF** (Trivial Graph Format), **Pajek** [14], **GraphML** (Graph Markup Language [15]), **GML** (Graph Modelling Language [16]), **GXL** (Graph eXchange Language [17]) представляют графы в виде списка ребер, а следовательно для решения поставленной задачи не подходят.

Единственным конструктивным методом, представленном в обзоре [11], является **NetML** [18]. Авторы обзора [11] относят его к конструктивно-процедурным. С помощью NetML кроме явного перечисления вершин и ребер графа можно использовать графы-звезды как элементы в описании более крупного графа. Также возможно применять преобразования разбиения ребер, слияния смежных ребер, и объединения ранее определенных подграфов. Данный язык слишком ограничен для наших целей, поскольку отсутствие параметризации и скромный набор преобразований не позволяет достаточно эффективно укоротить описания рассматриваемых графов: отсутствие включения сразу нескольких копий какого-либо подграфа, а также конструкций, упрощающих создание сразу нескольких ребер по некоторому правилу делают его непригодным для описания суперкомпьютерных сетей. Кроме того, при повторном включении некоего подграфа необходимо вручную переименовывать каждую включенную вершину для избежания конфликта имен. Также этот метод не предоставляет способа автоматического создания имен вершин.

2. Методы описания, используемые в системах моделирования сетей

Бóльшая часть систем моделирования и симуляции сетей использует процедурные методы описания сетей. Рассмотрим некоторые популярные системы.

ns-3 [19] является набором библиотек для симулирования коммуникационных сетей. Сети описываются только процедурным методом.

BigNetSim (BigSimulator, [20]) является симулятором коммуникационных сетей. Поддерживается ограниченное число топологий, и для описания собственных топологий требуется модификация исходного кода, что является процедурным методом описания сети.

mininet [21] — эмулятор, часто использующийся для моделирования программно-определяемых сетей. Он имеет ограниченное число встроенных топологий, и для определения собственной топологии предлагается использовать программный интерфейс на языке Python3, то есть процедурный метод.

INRFlow [22] имеет несколько встроенных топологий и позволяет загружать граф сети из внешнего файла в виде списка ребер.

OMNeT++ (Objective Modular Network Testbed in C++ [23, 24]) — библиотека и платформа для дискретно-событийного моделирования. Создана и используется для моделирования коммуникационных сетей, многопроцессорных систем и других распределенных вычислительных систем.

В OMNeT++ для описания сетей используется предметно-ориентированный язык NED (NEtwork Description).

Сеть состоит из вложенных модулей. Модули могут быть простыми (`simple`) или составными (`compound`). Модули соединяются друг с другом с помощью портов (`gates`) — абстрактных интерфейсов ввода-вывода модулей. Порты могут соответствовать портам ком-

мутаторов, сетевых карт или коммутационных панелей. Как подмодули, так и порты могут быть указаны в виде векторов.

Связи между модулями описываются в секции connections модулей. Для указания связей можно использовать циклы и условия. Это помогает в описании в том числе сложных структур с повторяющимися элементами и, в некоторой степени, нерегулярных структур. Пример описания сети на языке NED приведен на рис. 1.

```
simple Hub
  gates:
    out: outport[];
endsimple

simple Station
  gates:
    in: In;
endsimple

module Star
  submodules:
    hub: Hub
      gatesizes: outport[4];
    station: Station[4];
  connections:
    for i=0..3 do
      Hub.outport[i] --> Station[i].In;
    endfor
endmodule

network star: Star
endnetwork
```

Рис. 1. Описание сети на языке NED

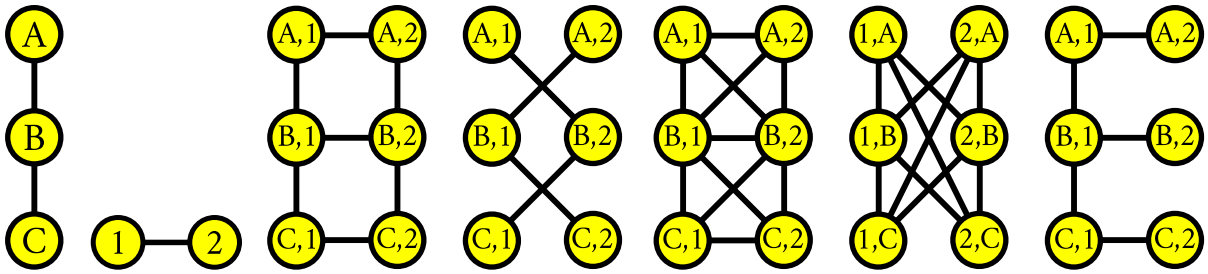
Можно заметить, что данный метод описания коммуникационной сети является конструктивным. Однако единственным преобразованием, реализованным в этом методе, является объединение графов.

3. Построение описания графов на основе топологических произведений

На основании обзора существующих методов был выбран конструктивный метод описания. За основу взята идея иерархично-модульной структуры, наподобие использующейся в OMNeT++. Такая структура также позволит создавать уникальное полное имя для каждой вершины графа. Но для более компактного описания метод должен поддерживать больше преобразований над подграфами, чем объединение.

Операции произведения графов могут использоваться при описания графов сетей (более подробно это обсуждается в [10]). Рассмотрим некоторые виды графовых произведений, которые могут использоваться с помощью предлагаемого метода.

Напомним, что декартовым произведением множеств называется множество, элементами которого являются все возможные упорядоченные пары элементов этих множеств. Пусть графы G и H имеют непересекающиеся множества вершин $V(G)$ и $V(H)$. Произ-



a) G b) H c) $G \square H$ d) $G \times H$ e) $G \times H$ f) $H \cdot G$ g) $G \circ H$

Рис. 2. Различные произведения графов

ведением графов G и H называется граф $G \times H$ такой, что $V(G \times H) = V(G) \times V(H)$ (множество вершин графа-произведения равно декартовому произведению множеств вершин графов-членов произведения). Множество ребер графа-произведения зависит от того, какой именно вид произведения рассматривается.

Было принято решение добавить поддержку декартового (или прямого, $G \square H$) [25], тензорного ($G \times H$) [26], лексикографического ($G \cdot H$) [27], сильного ($G \boxtimes H$) [25] и корневого ($G \circ H$) [28] произведений графов.

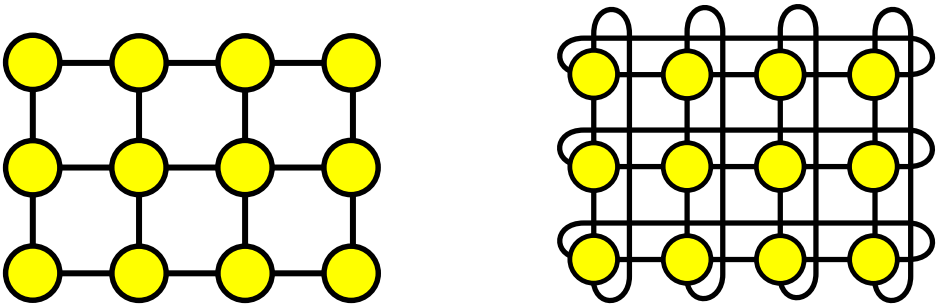
Две вершины называются смежными ($u \sim v$), если они соединены ребром (u, v).

Для всех видов произведений $P = G \times H$ графов G и H, где $e_g, f_g \in V(g)$ и $e_h, f_h \in V(H)$, две вершины $u = (e_g, e_h)$ и $v = (f_g, f_h)$ будут смежны при выполнении следующих условий:

- Для прямого произведения графов $(e_g, e_h) \sim (f_g, f_h) \Leftrightarrow e_g = f_g$ и $e_h \sim f_h$, либо $e_h = f_h$ и $e_g \sim f_g$.
- Для тензорного произведения $(e_g, e_h) \sim (f_g, f_h) \Leftrightarrow e_g \sim f_g$ и $e_h \sim f_h$.
- Для лексикографического произведения $(e_g, e_h) \sim (f_g, f_h) \Leftrightarrow e_g \sim f_g$ либо $e_g = f_g$ и $e_h \sim f_h$.
- Для сильного произведения — множество ребер равно объединению множеств ребер прямого и тензорного произведений.
- Для корневого произведения с заданным корнем $r_h \in H$ будет выполнено $(e_g, e_h) \sim (f_g, f_h) \Leftrightarrow e_g \sim f_g$, где $e_h = r_h$ и $f_h = r_h$, либо $e_g = f_g$ и $e_h \sim f_h$.

На рис. 2 изображены примеры различных видов графовых произведений. Если принять изображенный на рис. 2a граф за G, а изображенный на рис. 2b — за H, то граф, изображенный на рис. 2c, является прямым произведением G и H, изображенный на рис. 2d — тензорным, а граф, изображенный на рис. 2e, является одновременно и сильным, и лексикографическим произведением G и H. На рис. 2f изображен граф, являющийся лексикографическим произведением H и G, что демонстрирует некоммутативность лексикографического произведения. На рис. 2g изображен граф, являющийся корневым произведением G и H с выбранной в H корневой вершиной, помеченной цифрой 1.

Простыми для понимания примерами применения операций произведения для описания более сложных графов являются выражение решеток, торов и n-мерных кубов через прямое произведение графов-путей и графов-циклов. Подробнее про это написано в знаменитой книге [29]. На рис. 3 изображены графы решетки и тора. Проиллюстрированная решетка может быть выражена как прямое произведение графов P_3 и P_4, где P_n — граф-цепь длины n, а проиллюстрированный тор может быть выражен как прямое произведение C_3 и C_4, где C_n — граф-кольцо длины n. Несмотря на простоту приведенных примеров, мно-



a) $P_3 \times P_4$ b) $C_3 \times C_4$

Рис. 3. Решетка и тор, выраженные через прямые произведения

гомерные решетки и торы встречаются в топологиях суперкомпьютерных сетей (например, суперкомпьютерные системы серий IBM Blue Gene имеют сетевую топологию «многомерный тор» [30]), что демонстрирует возможное применение преобразования произведения подграфов в описании этих топологий.

Другим видом преобразования, которое может быть полезным для описания топологий суперкомпьютерных сетей, является замена одного подграфа на другой. Поскольку реальные системы зачастую имеют «дефекты», или «нерегулярности», связанные с тем, что либо суперкомпьютерную систему планируется расширять в будущем, либо существующую систему с регулярной топологией уже расширили — следует предусмотреть способ корректного описания таких случаев.

Наконец, можно упростить три простых, но часто встречающихся сюжета: соединение в некотором подмножестве вершин графа каждой вершины с каждой, и соединение вершин некоторого подграфа в цепь или в кольцо.

3.1. Язык описания топологий

Было решено создать текстовый формат (для удобства чтения и написания человеком), основанный на JSON [31] (для удобства разбора в программных средствах). Пример описания полного графа в предлагаемом формате представлен на рис. 4.

```
1 [{ "simplemodule": { "name": "simple" }},
2 { "module": {
3     "name": "alltoall",
4     "submodules": [ { "name": "node",
5                       "module": "simple",
6                       "size": "10" }],
7     "connections": [{ "all-match": "node" }] }},
8 { "network": { "module": "alltoall" }}]
```

Рис. 4. Пример описания полного графа в предлагаемом формате

За базовую единицу построения был выбран *модуль* — некоторый подграф. Сеть (`network`) состоит из одного модуля. Модули разделяются на *простые* (`simplemodule`) и *составные* (`module`). Одному простому модулю соответствует одна вершина в графовом представлении.

Задача составного модуля заключается в объединении подграфов, составляющих несколько модулей, в некоторый больший граф. Составные модули могут иметь внутри

несколько составных подмодулей, которые в конечном итоге будут состоять из простых модулей.

Модули соединяются друг с другом с помощью *портов* (`gates`). Как было упомянуто ранее, «порт» является абстрактным понятием: порты простых модулей могут соответствовать портам коммутаторов либо сетевых карт вычислительных узлов, а порты составных модулей могут соответствовать портам коммутационных панелей. Порты простых модулей напрямую связаны с вершиной, соответствующей этому модулю и могут соединяться не более чем с одним другим портом. Порты составных модулей могут соединяться не более чем с двумя другими портами.

Кроме включения в модуль подмодулей, указанных перечислением, в модуль можно включить сразу несколько подмодулей одного типа, включив *вектор* подмодулей. Параметр `size` определяет размер вектора. В таком случае в модуль добавляется несколько подмодулей с именем `name[i]`, где `name` — название, с которым был добавлен данный подмодуль. Каждому подмодулю, являющемуся частью вектора, доступен параметр `index`, который равен индексу подмодуля в векторе (про параметры будет подробно рассказано далее). С помощью механизма векторов подмодулей значительно упрощается задача описания повторяющихся элементов сетевой топологии.

Аналогично в виде вектора могут описываться порты, принадлежащие какому-либо модулю. Применим тот же принцип именования результирующих портов — `name[i]`. Таким образом упрощаются описания, например, коммутаторов с большим числом портов.

Подмодули могут иметь *параметры* (`params`). Параметры используются при вычислении значения арифметических выражений. Арифметические выражения могут встречаться в нескольких местах:

- при описании векторов подмодулей или векторов портов размер данных векторов (`size`) задается арифметическим выражением;
- при описании соединения — если соединяемый модуль является частью вектора; таким образом, в описание некоего соединения может входить, например, не `node[3]`, а `node[i]`, если этого требует конкретная топология;
- также арифметическим выражением может задаваться индекс порта, если указанный порт является частью вектора портов;
- при условном включении подмодулей и при условном описании ребер арифметическим выражением задается выражение, определяющее истинность условия (про условное включение подмодулей и описание ребер подробнее рассказано далее);
- при использовании циклов для описания ребер графа арифметическим выражением задаются начальное и конечное значения переменной цикла (про описание ребер в цикле подробнее рассказано далее).

Параметры указываются при определении сети, при определении модуля, и при включении подмодулей. При наличии нескольких определений параметра с одним и тем же именем более приоритетными считаются определения, находящиеся «глубже» (например, параметры, определенные в данном модуле, приоритетнее, чем параметры, определенные для всей сети).

Для построения составного модуля подграфы, соответствующие его подмодулям, могут либо объединяться, либо входить в виде произведения (`cartesian`, `tensor`, `lexicographical`, `strong`, `rooted`). При описании `rooted` – корневого произведения, требуется указать корень — `root`.

Подмодули также могут входить в модуль условно — с помощью условного оператора (`if: ..., then: ..., else: ...`). Таким образом можно, например, описывать граничные условия рекурсивных структур.

Модули соединяются друг с другом с помощью портов. Однако при желании порты можно опустить и при указании ребер соединять простые модули напрямую; в таком случае создаются порты `_auto[n]`, соединенные с указанными вместо портов простыми модулями, где n — следующий незанятый порт, начиная с 0. Такой вид соединения может быть удобен в случаях, когда вычислять номер следующего свободного порта в векторе портов, связанных с некоторым простым модулем, трудоемко, и конкретный номер порта для каждого соединения не представляет важности. Нумерация автоматически созданных портов для каждого конкретного описания всегда одна и та же; принимаем, что соединения создаются в рамках каждого модуля последовательно в порядке их указания.

Перед применением операций произведения над подграфами в обязательном порядке производится операция *сжатия* — удаления всех портов, кроме имеющих лишь одно инцидентное ребро. Эта операция производится потому, что цепи, проходящие от одного простого модуля к другому через некоторую последовательность портов, соответствуют одному ребру в результирующем графе, и потому порты в операциях произведения участвовать не должны. «Висячие» порты, связанные лишь с одним простым модулем, в графе-результате произведения присутствуют у каждой вершины, являющейся результатом произведения исходного простого модуля с какой-либо вершиной второго графа. На рис. 5 проиллюстрирован результат тензорного произведения двух подмодулей. Порты, связывающие узлы в исходных подмодулях, не являются частью результирующего модуля, но оставшиеся свободными порты — являются.

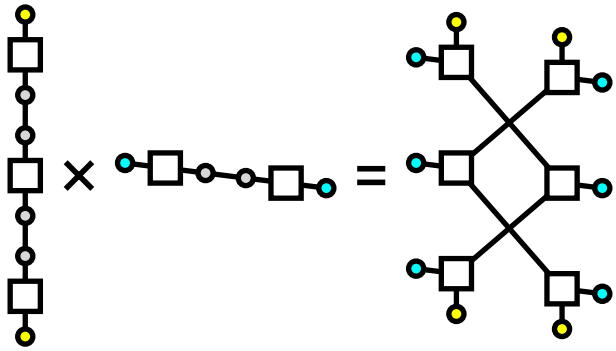


Рис. 5. Тензорное произведение двух подмодулей

При необходимости исключить порты из итогового графа можно также производить операцию сжатия.

Ребра могут указываться как простым перечислением (`from: ..., to: ...,`), так и в цикле по некоторой переменной (`loop: ..., start: ..., end: ..., conn: ...`). Также ребра могут указываться условно (`if: ..., then: ..., else: ...`). Кроме того, перечислив в виде цикла список простых модулей, возможно соединить их в линию (`line`), кольцо (`ring`), или каждую со каждой (`all`). Также возможно соединить набор вершин по принципу «каждая с каждой», задав его расширенным регулярным выражением в формате POSIX [32], поскольку порядок перечисления вершин в полносвязном графе не важен (`all-match`).

Для описания «нерегулярностей» в описание модуля можно включить замещение части вложенных в него вершин и портов, описанных также расширенным регулярным выражением POSIX, другим подмодулем (`replace: nodes: ..., with: ...`). Заменой части вершин на пустой модуль можно отразить такую нерегулярность, как отсутствие узлов.

Каждый простой модуль и каждое соединение может иметь связанную с ними дополнительную информацию (`attributes`). С помощью дополнительной информации можно, например, указать пропускную способность соединений, или отличить маршрутизаторы от вычислительных узлов.

Имена всех модулей и портов создаются исходя из иерархической структуры сети. Именем каждого модуля является конкатенация через точку (`.`) имен всех модулей, в которые он входит, от внешнего к внутреннему, и имени самого модуля. Именем каждого порта является конкатенация через точку имени модуля, которому он принадлежит, и имени порта. Таким образом каждый узел и каждый порт имеет автоматически созданное имя, напрямую следующее из положения в структуре сети.

4. Примеры описаний

4.1. Описание решетки

```
[{ "simplemodule": { "name": "grid_simple" }},
{ "module": { "name": "grid_ring",
    "submodules": [{ "name":"s", "module":"grid_simple", "size":"len"}],
    "connections": [{
            "line": "i",
            "start": "0",
            "end": "len",
            "conn": "s[i]",
            "attributes": "style=dashed" }]}},
{ "module": { "name": "grid_prod",
    "submodules": [ {
        "if": "depth == 1",
        "then": { "name": "a", "module": "grid_ring" },
        "else": {
            "cartesian": [
                { "name": "a", "module": "grid_ring" },
                { "if": "depth > 2", "then": {
                        "name": "p",
                        "module": "grid_prod",
                        "params": [ { "depth": "depth - 1" } ]
                }, "else": {
                        "name": "b", "module": "grid_ring" }}]}}]}},
{ "network": {
    "module": "grid_prod",
    "params": [ { "len": "3" }, { "depth": "3" } ] }}]
```

Рис. 6. Пример описания многомерной решетки в предлагаемом формате

Как было упомянуто ранее, многомерная решетка может быть представлена как прямое произведение нескольких простых цепей. Это свойство используется в данном описании для предоставления параметризации топологии не только по длине, но и по числу измере-

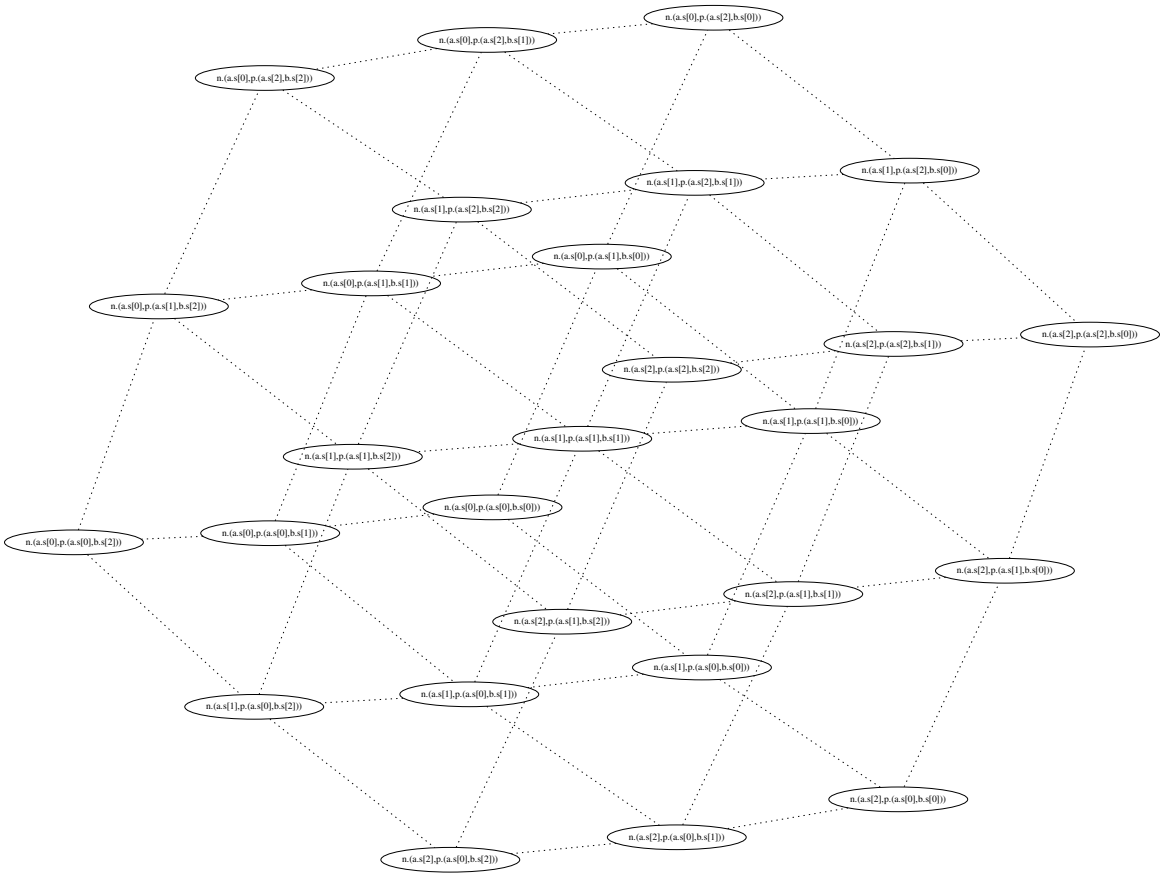


Рис. 7. Результирующий граф многомерной решетки

ний. Решетка описана рекурсивно, и граничное условие рекурсии описано оператором `if`. Параметр `depth` на каждом шаге рекурсии уменьшается на единицу, пока не достигает значения 1. Базовая цепь описывается с помощью оператора `line`.

Пример описания многомерной решетки приведен в листинге 6, результирующий граф проиллюстрирован на рис. 7 [1]. В иллюстративных целях каждой связи было придано свойство `style=dashed`, которое после конвертирования в формат GraphViz DOT было преобразовано в соответствующее свойство ребер и далее использовано программой визуализации графов для задания пунктирного вида ребер.

4.2. Пример описания жирного дерева (Fat tree)

Пример описания утолщенного дерева с нерегулярностями приведен в листинге 8, а результирующий граф проиллюстрирован на рис. 9. Сама топология утолщенного дерева используется многими суперкомпьютерными системами, в том числе в находящимися на самом верху списка TOP 500 самых производительных вычислительных систем [1] — Summit и Sierra [33], Tianhe-2A [34], AI Bridging Cloud Infrastructure [35], и многими другими.

В данном примере узлы, отражающие маршрутизаторы, и отражающие вычислительные узлы, описаны разными простыми модулями, что позволяет различить их с помощью указания дополнительной информации в поле `attributes`. Обобщенное утолщенное дерево

[1] Узлы решетки на рисунке имеют имена вида `n.(a.s[1],p.(a.s[2],b.s[1]))`, в соответствии с указанными именами простого и составных модулей, и именем сети по умолчанию в используемой реализации.

```
[{ "simplemodule": { "name": "gft_router" }},
{ "simplemodule": { "name": "gft_endnode", "attributes": "shape=box" }},
{ "module": { "name": "empty" }},
{ "module": {
    "name": "gft_layer",
    "submodules": [{
        "if": "h == 2", "then": {
            "name": "node", "module": "gft_endnode", "size": "w^h"
        }, "else": {
            "name": "node", "module": "gft_router", "size": "w^h"
    }}, { "if": "h > 0", "then": {
              "name": "prev", "module": "gft_layer", "size": "m",
              "params": [ { "h": "h-1" } ] }}],
    "connections": [{
        "if": "h > 0", "then": {
            "loop": "i", "start": "0", "end": "w^h",
            "conn": {
                "loop": "j", "start": "0", "end": "m",
                "conn": {
                    "from": "prev[j].node[floor(i/w)]",
                    "to": "node[i]" }}}}] }},
{ "module": {
    "name": "gft_alltoall",
    "submodules": [{"name":"node", "module":"gft_endnode", "size":"3"}],
    "connections": [{ "all-match": "gft.node\\[[0-2]\\]$" }] }},
{ "module": {
    "name": "gft_m",
    "submodules": [{
        "name": "gft", "module": "gft_layer",
        "params": [ { "h": "2" }, { "m": "2" }, { "w": "3" } ] }],
    "replace": [{
        "nodes": "gft.node\\[[0-2]\\]$",
        "with": { "name": "gft", "module": "gft_alltoall" } }, {
        "nodes": "gft.node\\[8\\]$",
        "with": { "name": "e", "module": "empty" } }]}},
{ "network": { "module": "gft_m" }}]
```

Рис. 8. Пример описания обобщенного утолщенного дерева с нерегулярностями в предлагаемом формате

описано с помощью модуля `gft_layer`. Ему передаются три параметра: `h` — высота дерева, `m` — число связей, уходящих «вверх», и `w` — число связей, уходящих «вниз». Более подробно про обобщенные утолщенные деревья описано в [4].

В демонстрационных целях приведенная сеть имеет две нерегулярности: в одной из ветвей утолщенного дерева три узла связаны по принципу «все со всеми», а в другой ветви отсутствует один из узлов. Отсутствие одного из узлов описано с помощью его замены пустым модулем (в данном примере — `empty`).

4.3. Описание Ломоносов-2

Наконец, в листинге 10 приведено описание суперкомпьютера Ломоносов-2, установленного в МГУ им. М.В. Ломоносова. Подробнее система описана в [36, 37].

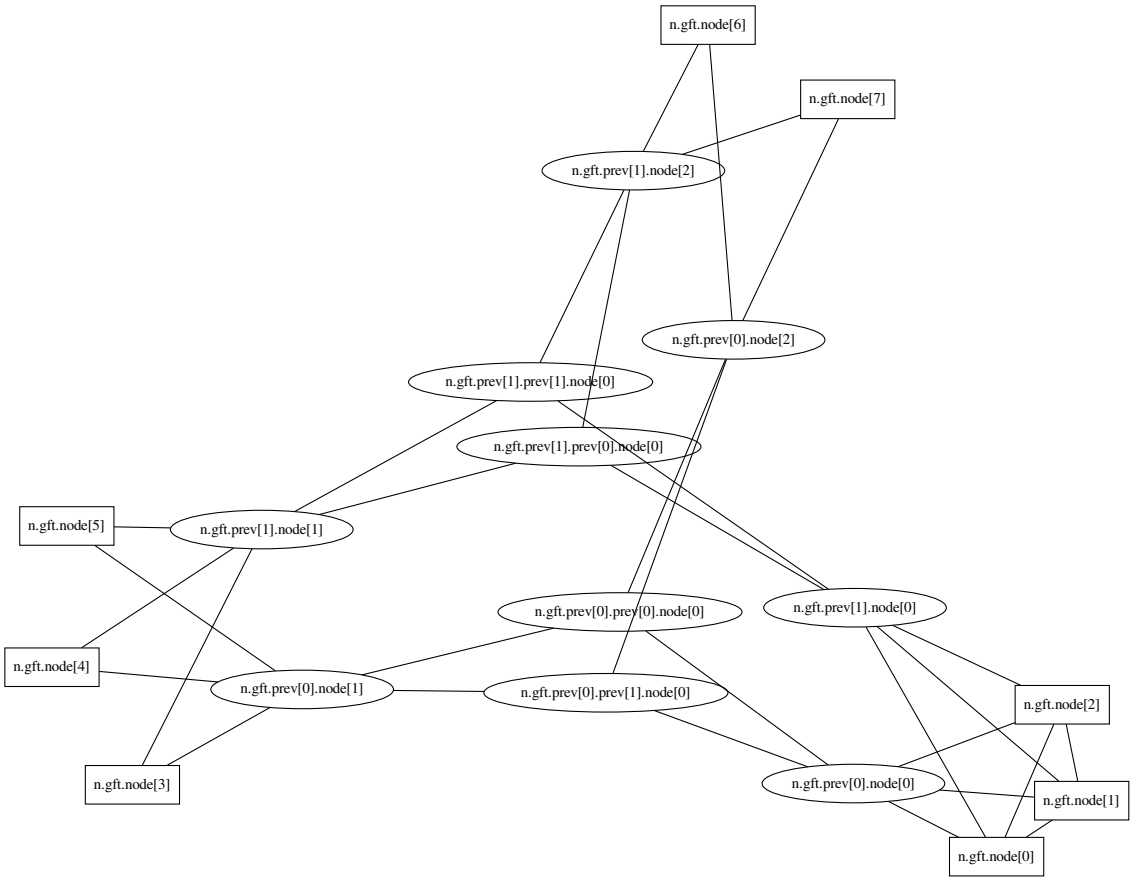


Рис. 9. Результирующий граф нерегулярного утолщенного дерева

В приведенном описании учтены нерегулярности, присутствующие в данном суперкомпьютере. Хотя топология суперкомпьютера основана на «плоской бабочке» размера $4\times8\times8$, к каждому маршрутизатору которой подключено по 8 узлов, на данный момент в системе лишь 1696 узлов; «плоская бабочка» имеет размеры $4\times8\times7$, и дополнительно отсутствуют 12 коммутаторов.

5. Сравнение размеров описаний графов

Для оценки компактности сравним описания некоторых топологий с помощью предложенного метода и с помощью GraphViz DOT (табл. 1).

Поскольку одно и то же описание, за исключением заданных параметров, может использоваться для нескольких частных случаев одной и той же топологии, значения в последней колонке повторяются.

Как видно, предлагаемый метод более краткий, чем GraphViz DOT, и разница более значительна для больших и регулярных сетей. Это объясняется тем, что после описания параметризованной топологии в предложенном формате размер описания не меняется при изменении его параметров, в то время как размер описания в формате GraphViz DOT неизбежно растет.

```
[{ "simplemodule": { "name": "gft_router" }},
{ "simplemodule": { "name": "gft_endnode", "attributes": "shape=box" }},
{ "module": { "name": "empty" }},
{ "module": {
    "name": "gft_layer",
    "submodules": [{
        "if": "h == 2", "then": {
            "name": "node", "module": "gft_endnode", "size": "w^h"
        }, "else": {
            "name": "node", "module": "gft_router", "size": "w^h"
    }}, { "if": "h > 0", "then": {
              "name": "prev", "module": "gft_layer", "size": "m",
              "params": [ { "h": "h-1" } ] }}],
    "connections": [{
        "if": "h > 0", "then": {
            "loop": "i", "start": "0", "end": "w^h",
            "conn": {
                "loop": "j", "start": "0", "end": "m",
                "conn": {
                    "from": "prev[j].node[floor(i/w)]",
                    "to": "node[i]" }}}}] }},
{ "module": {
    "name": "gft_alltoall",
    "submodules": [{"name":"node", "module":"gft_endnode", "size":"3"}],
    "connections": [{ "all-match": "gft.node\\[[0-2]\\]$" }] }},
{ "module": {
    "name": "gft_m",
    "submodules": [{
        "name": "gft", "module": "gft_layer",
        "params": [ { "h": "2" }, { "m": "2" }, { "w": "3" } ] }],
    "replace": [{
        "nodes": "gft.node\\[[0-2]\\]$",
        "with": { "name": "gft", "module": "gft_alltoall" } }, {
        "nodes": "gft.node\\[8\\]$",
        "with": { "name": "e", "module": "empty" } }]}},
{ "network": { "module": "gft_m" }}]
```

Рис. 10. Пример описания суперкомпьютера Ломоносов-2 в предлагаемом формате

6. Библиотека для работы с топологиями в предложенном формате

Для работы с форматом была разработана библиотека, написанная на языке Си, не имеющая внешних зависимостей, за исключением стандартной библиотеки. Такое решение было принято для обеспечения большей переносимости между UNIX системами. Библиотека предоставляет следующие функции:

- `topologies_network_init` — производит инициализацию структуры, представляющей сеть;
- `topologies_network_read_file` — производит чтение из файла;
- `topologies_network_read_string` — производит чтение из строки;
- `topologies_definition_to_graph` — вычисляет графовое представление сети;
- `topologies_graph_compact` — производит операцию сжатия графа;

Таблица 1. Сравнение с DOT

Топология		DOT, байт	Предлагаемый метод, байт	Отношение, %
Полный граф K_n	K_5	180	162	111%
	K_{10}	525	163	322%
	K_{20}	1 950	163	1.2%
Многомерная решетка	2x2x2x2	1 095	493	222%
	3x3x3	1 581	493	320%
	4x4x4	3 942	493	800%
Многомерный тор	2x2x2x2	1 095	493	222%
	3x3x3	1 803	493	366%
	4x4x4	4 356	493	884%
Утолщенное дерево $GFT(h, m, w)$	$GFT(2, 3, 1)$	554	421	132%
	$GFT(2, 2, 3)$	796	421	189%
	$GFT(3, 4, 3)$	11 058	421	2.6%
Плоская бабочка	2-ary 4-flat	246	382	64%
	4-ary 3-flat	698	382	183%
	4-ary 4-flat	3 836	382	1.0%
Tianhe-2A	Утолщенное дерево [34]	1 192 132	1 152	103.5%
AI Bridging Cloud Infrastructure	Утолщенное дерево [35]	71 749	1 002	7.2%
Ломоносов-2	Плоская бабочка $4 \times 8 \times 8$ [36]	143 539	1 258	11.4%

- `topologies_graph_string` — возвращает представление графа в формате GraphViz DOT;
- `topologies_graph_string_free` — освобождает память, выделенную под результирующую строку;
- `topologies_graph_print` — выводит представление графа в формате GraphViz DOT в файловый дескриптор;
- `topologies_graph_destroy` — освобождает память, выделенную под граф;
- `topologies_network_destroy` — освобождает память, выделенную под сеть.

Также было разработано программное средство на языке Python3 для визуализации графовой структуры описаний. Программе передается на вход один или несколько файлов, содержащих описание одной сети. Программа отображает интерактивное окно с визуализацией графа.

Для работы с описанной выше библиотекой, написанной на языке Си, из программного средства на языке Python3, был использован модуль ctypes из стандартной поставки Python3. С помощью этого модуля была написана обертка, предоставляющая интерфейс к библиотеке. Используя вышеописанную обертку основной код визуализатора конвертирует представление в формат GraphViz DOT, а затем передает полученный граф библиотеке graph_tool, которая и предоставляет функции интерактивной визуализации.

Оберткой для Python3 предоставляются следующие функции:

- `Topologies.__init__(self, definition)` — создает объект Python для работы с описанием сети. Параметр `definition` задает описание в виде строки.
- `Topologies.network_parse(self, compress, print_gates)` — производит работу с библиотекой и возвращает представление графа в формате GraphViz DOT в виде строки. При возникновении ошибки во время работы с библиотекой порождает исключение типа `ValueError`, содержащее строку с подробным описанием ошибки. Булевый параметр `compress` определяет, необходимо ли производить операцию сжатия (удаления всех портов, связанных с двумя вершинами графа). Булевый параметр `print_gates` определяет, включать ли порты, связанные менее чем с двумя вершинами графа, в итоговый граф.

Заключение

Итак, предложенный подход решает сложную задачу описания топологий сетей, которые носят в основном регулярную/симметричную структуру. Классические методы описания графов в данной ситуации не эффективны, поскольку приводят к громадным описаниям по сути одних и тех же элементов. Нами в статье был представлен метод описания графов топологий коммуникационных сетей суперкомпьютерных систем:

- являющийся компактным для регулярных структур, в том числе даже если эти структуры не совершенно идеальны;
- позволяющий именовать узлы сети исходя из описания топологии;
- предположительно, также подходящий для чтения и написания человеком (однако подтверждение этого факта требует дополнительного исследования и является возможным направлением развития данной работы).

Также были разработаны программные средства — библиотека для работы с разработанным форматом, и программа для визуализации описаний.

Метод был проверен и сравнен с GraphViz DOT на описаниях некоторых известных топологий и на описаниях существующих суперкомпьютерных систем, и оказался более кратким, особенно для больших регулярных сетей. При сравнении описаний систем Tianhe-2A, AI Bridging Cloud Infrastructure и Ломоносов-2 разница составила соответственно 103.484%, 7.160% и 11.410%.

Созданный программные коды библиотеки на языке Си и обертки над ней на языке Python3 находится в открытом доступе по ссылке `https://github.com/asalnikov/topologies/`. В том же репозитории можно найти формальное описание грамматики предлагаемого формата.

Литература

1. June 2022 | TOP500 Supercomputer Sites. URL: `https://top500.org/lists/top500/2022/06/` (дата обращения: 21.08.2022).
2. Воеводин В.В., Воеводин В.В. Параллельные вычисления. БХВ-Петербург, 2002. 608 с.
3. Leiserson C.E. Fat-trees: Universal networks for hardware-efficient supercomputing // IEEE Transactions on Computers. 1985. Vol. 34, no. 10. P. 892–901. DOI: `10.1109/TC.1985.6312192`.

4. Ohring S.R., Ibel M., Das S.K., Kumar M.J. On generalized fat trees // Proceedings of IPPS '95, The 9th International Parallel Processing Symposium, Santa Barbara, California, USA, April 25-28, 1995. IEEE, 1995. P. 37–44. DOI: 10.1109/IPPS.1995.395911.

5. Kim J., Dally W.J., Scott S., Abts D. Technology-Driven, Highly-Scalable Dragonfly Topology // 35th International Symposium on Computer Architecture (ISCA 2008), Beijing, China, June 21-25, 2008. IEEE, 2008. P. 77–88. DOI: 10.1109/ISCA.2008.19.

6. Kim J., Dally W.J., Abts D. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks // 34th International Symposium on Computer Architecture (ISCA 2007), San Diego, California, USA, June 9-13, 2007. ACM, 2007. P. 126–137. DOI: 10.1145/1250662.1250679.

7. Petrini F., Kerbyson D.J., Pakin S. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q // Proceedings of the ACM/IEEE SC2003 Conference on High Performance Networking and Computing, Phoenix, AZ, USA, November 15-21, 2003, CD-Rom. 2003. P. 55–55. DOI: 10.1145/1048935.1050204.

8. Zheng G., Wilmarth T., Jagadishprasad P., Kalé L. Simulation-Based Performance Prediction for Large Parallel Machines // International Journal of Parallel Programming. 2005. Vol. 33, no. 2-3. P. 183–207. DOI: 10.1007/s10766-005-3582-6.

9. Liu N., Carothers C., Cope J., *et al.* Model and Simulation of Exascale Communication Networks // Journal of Simulation. 2012. Vol. 6, no. 4. P. 227–236. DOI: 10.1057/jos.2012.4.

10. Parsonage E., Nguyen H.X., Bowden R., *et al.* Generalized graph products for network design and analysis // Proceedings of the 19th annual IEEE International Conference on Network Protocols, ICNP 2011, Vancouver, BC, Canada, October 17-20, 2011. IEEE, 2011. P. 79–88. DOI: 10.1109/ICNP.2011.6089084.

11. Roughan M., Tuke S.J. Unravelling Graph-Exchange File Formats // CoRR. 2015. Vol. abs/1503.02781. arXiv: 1503.02781. URL: http://arxiv.org/abs/1503.02781.

12. Siek J., Lee L.-Q., Lumsdaine A. The Boost Graph Library: User Guide and Reference Manual. 2002. 346 p.

13. Ellson J., Gansner E.R., Koutsofios E., *et al.* Graphviz – Open Source Graph Drawing Tools // Graph Drawing, 9th International Symposium, GD 2001 Vienna, Austria, September 23-26, 2001, Revised Papers. Vol. 2265 / ed. by P. Mutzel, M. Jünger, S. Leipert. Springer, 2001. P. 483–484. Lecture Notes in Computer Science. DOI: 10.1007/3-540-45848-4_57.

14. Batagelj V., Mrvar A. Pajek – Program for Large Network Analysis. 1999.

15. Tamassia R. Handbook of Graph Drawing and Visualization. 1st. Chapman & Hall/CRC, 2016.

16. Himsolt M. GML: A portable Graph File Format. 2010. URL: http://www.fim.uni-passau.de/fileadmin/files/lehrstuhl/brandenburg/projekte/gml/gml-technical-report.pdf.

17. Holt R.C., Schürr A., Sim S.E., Winter A. GXL: A graph-based standard exchange format for reengineering // Science of Computer Programming. 2006. Vol. 60, no. 2. P. 149–170. DOI: 10.1016/j.scico.2005.10.003.

18. Batagelj V., Mrvar A. Towards NetML Networks Markup Language // International Social Network Conference, London. 1995.

19. Kumar A.R.A., Rao S.V., Goswami D. NS3 Simulator for a Study of Data Center Networks // IEEE 12th International Symposium on Parallel and Distributed Computing, ISPDC 2013, Bucharest, Romania, June 27-30, 2013. IEEE, 2013. P. 224–231. DOI: 10.1109/ISPDC.2013.37.

20. Choudhury N., Mehta Y., Wilmarth T.L., *et al.* Scaling an Optimistic Parallel Simulation of Large-Scale Interconnection Networks // Proceedings of the 37th Winter Simulation Conference, Orlando, FL, USA, December 4-7, 2005. IEEE, 2005. P. 591–600. DOI: 10.1109/WSC.2005.1574299.

21. Lantz B., Heller B., McKeown N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks // Proceedings of the 9th ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA, October 20-21, 2010. ACM, 2010. Article 19. DOI: 10.1145/1868447.1868466.

22. Navaridas J., Pascual J.A., Erickson A., *et al.* INRFlow: An interconnection networks research flow-level simulation framework // Journal of Parallel and Distributed Computing. 2019. Vol. 130. P. 140–152. DOI: 10.1016/j.jpdc.2019.03.013.

23. Varga A. The OMNET++ discrete event simulation system // Proc. ESM'2001. 2001. Vol. 9.

24. Varga A., Hornig R. An Overview of the OMNeT++ Simulation Environment // Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, SimuTools 2008, Marseille, France, March 3-7, 2008. ICST/ACM, 2008. Article 60. DOI: 10.5555/1416222.1416290.

25. Sabidussi G. Graph multiplication // Mathematische Zeitschrift. 1959. Vol. 72, no. 1. P. 446–457. DOI: 10.1007/bf01162967.

26. Weichsel P. The Kronecker Product of Graphs // Proceedings of The American Mathematical Society - PROC AMER MATH SOC. 1962. Vol. 13. DOI: 10.2307/2033769.

27. Geller D., Stahl S. The chromatic number and other functions of the lexicographic product // Journal of Combinatorial Theory, Series B. 1975. Vol. 19, no. 1. P. 87–95. DOI: 10.1016/0095-8956(75)90076-3.

28. Godsil C., McKay B. A new graph product and its spectrum // Bulletin of The Australian Mathematical Society - BULL AUST MATH SOC. 1978. Vol. 18. DOI: 10.1017/S0004972700007760.

29. Кнут Д. Искусство программирования, том 4, А. Комбинаторные алгоритмы, часть 1. Вильямс, 2013.

30. Chen D., Eisley N.A., Heidelberger P., *et al.* The IBM Blue Gene/Q Interconnection Network and Message Unit // Conference on High Performance Computing Networking, Storage and Analysis, SC 2011, Seattle, WA, USA, November 12-18, 2011. ACM, 2011. Article 26. DOI: 10.1145/2063384.2063419.

31. Bray T. The JavaScript Object Notation (JSON) Data Interchange Format: RFC / RFC Editor. 2017. No. 8259. URL: https://www.rfc-editor.org/rfc/rfc8259.txt.

32. IEEE Standard for Information Technology–Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7 // IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008). 2018. P. 1–3951. DOI: 10.1109/IEEESTD.2018.8277153.

33. Stunkel C.B., Graham R.L., Shainer G., *et al.* The high-speed networks of the Summit and Sierra supercomputers // IBM J. Res. Dev. 2020. Vol. 64, no. 3/4. 3:1–3:10. DOI: 10.1147/JRD.2020.2967330.

34. Liao X.-K., Pang Z.-B., Wang K.-F., *et al.* High Performance Interconnect Network for Tianhe System // Journal of Computer Science and Technology. 2015. Vol. 30, no. 2. P. 259–272. DOI: 10.1007/s11390-015-1520-7.

35. Takizawa S. AI Bridging Cloud Infrastructure (ABCI) and its Communication Performance. 2019. URL: http://mug.mvapich.cse.ohio-state.edu/mug/19/ 7th Annual MVAPICH User Group Meeting.

36. Voevodin V., Antonov A., Nikitenko D., *et al.* Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community // Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

37. PARALLEL.RU. Суперкомпьютер “ЛОМОНОСОВ-2” | PARALLEL.RU – Информационно-аналитический центр по параллельным вычислениям. URL: https://parallel.ru/cluster/lomonosov2.html (дата обращения: 15.05.2020).

Хабирова Эльвира Радмировна, инженер, ООО «Открытая Мобильная Платформа» (Москва, Российская Федерация)

Сальников Алексей Николаевич, к.ф.-м.н., ведущий научный сотрудник, Московский государственный университет имени М.В. Ломоносова (Москва, Российская Федерация), Федеральный исследовательский центр «Информатика и управление» РАН (Москва, Российская Федерация)

DOI: 10.14529/cmse220302

TOPOLOGICAL STRUCTURE OF COMPUTING CLUSTERS DESCRIPTION METHOD BASED ON OPERATIONS OF SUBGRAPHS MULTIPLICATION

© 2022 E.R. Khabirova[1], A.N. Salnikov[2,3]

[1]*LLC "Open Mobile Platform" (pr. Vernadskogo 41, Moscow, 119415 Russia),*

[2]*Lomonosov Moscow State University (GSP-1, Leninskie Gory 1, Moscow, 119991 Russia),*

[3]*Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences (Vavilova 44-2, Moscow, 119333 Russia)*

E-mail: lineprinter0@gmail.com, salnikov@cs.msu.ru

Received: 22.08.2022

Topological structure of communication networks in supercomputers with grow in size and complexity of installation, respectively becomes more difficult. There are many methods to describe it, but such descriptions are cumbersome, which makes them difficult to manipulate. The article proposes an approach to describing the communication environment of a supercomputer, when the communication network is described as a constructor. The elements of the constructor are typical topological structures often found in various computing systems. For this purpose, a language for describing the topological structure has been developed. It based on the operation products of subgraphs. The language is ideologically similar in its principles to the NetML and OMNeT++ languages. Special attention is paid to exceptions in the regularity of networks of real supercomputers; in order to add the possibility of describing this fact, special constructions have been introduced into the language. A library has been developed in the C programming language with purpose to facilitate work with the language intoduced in this article. Also a special wrapper over C library has been written in Python3, which then can be used to visualize graphs described by the language. The expressive power of language has been demonstrated in the description computing clusters: Tianhe-2A, AI Bridging Cloud Infrastructure and Lomonosov-2. The method has been tested and compared with GraphViz DOT it is showed multiple reductions in the Record volume required to save topology for some of the major Top500 systems.

Keywords: compute cluster, network topology, graph describing languages, subgraphs multiplications.

FOR CITATION

Khabirova E.R., Salnikov A.N. Topological Structure of Computing Clusters Description Method Based on Operations of Subgraphs Multiplications. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 3. P. 22–44. (in Russian) DOI: 10.14529/cmse220302.

References

1. June 2022 | TOP500 Supercomputer Sites. URL: `https://top500.org/lists/top500/2022/06/` (accessed: 21.08.2022).
2. Voevodin V.V., Voevodin V.V. Parallel Computing. BHV-Petersburg, 2002. 608 p.
3. Leiserson C.E. Fat-trees: Universal networks for hardware-efficient supercomputing. IEEE Transactions on Computers. 1985. Vol. 34, no. 10. P. 892–901. DOI: `10.1109/TC.1985.6312192`.

4. Ohring S.R., Ibel M., Das S.K., Kumar M.J. On generalized fat trees. Proceedings of IPPS '95, The 9th International Parallel Processing Symposium, Santa Barbara, California, USA, April 25-28, 1995. IEEE, 1995. P. 37–44. DOI: 10.1109/IPPS.1995.395911.

5. Kim J., Dally W.J., Scott S., Abts D. Technology-Driven, Highly-Scalable Dragonfly Topology. 35th International Symposium on Computer Architecture (ISCA 2008), Beijing, China, June 21-25, 2008. IEEE, 2008. P. 77–88. DOI: 10.1109/ISCA.2008.19.

6. Kim J., Dally W.J., Abts D. Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks. 34th International Symposium on Computer Architecture (ISCA 2007), San Diego, California, USA, June 9-13, 2007. ACM, 2007. P. 126–137. DOI: 10.1145/1250662.1250679.

7. Petrini F., Kerbyson D.J., Pakin S. The Case of the Missing Supercomputer Performance: Achieving Optimal Performance on the 8,192 Processors of ASCI Q. Proceedings of the ACM/IEEE SC2003 Conference on High Performance Networking and Computing, Phoenix, AZ, USA, November 15-21, 2003, CD-Rom. 2003. P. 55–55. DOI: 10.1145/1048935.1050204.

8. Zheng G., Wilmarth T., Jagadishprasad P., Kalé L. Simulation-Based Performance Prediction for Large Parallel Machines. International Journal of Parallel Programming. 2005. Vol. 33, no. 2-3. P. 183–207. DOI: 10.1007/s10766-005-3582-6.

9. Liu N., Carothers C., Cope J., *et al.* Model and Simulation of Exascale Communication Networks. Journal of Simulation. 2012. Vol. 6, no. 4. P. 227–236. DOI: 10.1057/jos.2012.4.

10. Parsonage E., Nguyen H.X., Bowden R., *et al.* Generalized graph products for network design and analysis. Proceedings of the 19th annual IEEE International Conference on Network Protocols, ICNP 2011, Vancouver, BC, Canada, October 17-20, 2011. IEEE, 2011. P. 79–88. DOI: 10.1109/ICNP.2011.6089084.

11. Roughan M., Tuke S.J. Unravelling Graph-Exchange File Formats. CoRR. 2015. Vol. abs/1503.02781. arXiv: 1503.02781. URL: http://arxiv.org/abs/1503.02781.

12. Siek J., Lee L.-Q., Lumsdaine A. The Boost Graph Library: User Guide and Reference Manual. 2002. 346 p.

13. Ellson J., Gansner E.R., Koutsofios E., *et al.* Graphviz – Open Source Graph Drawing Tools. Graph Drawing, 9th International Symposium, GD 2001 Vienna, Austria, September 23-26, 2001, Revised Papers. Vol. 2265 / ed. by P. Mutzel, M. Jünger, S. Leipert. Springer, 2001. P. 483–484. Lecture Notes in Computer Science. DOI: 10.1007/3-540-45848-4_57.

14. Batagelj V., Mrvar A. Pajek – Program for Large Network Analysis. 1999.

15. Tamassia R. Handbook of Graph Drawing and Visualization. 1st. Chapman & Hall/CRC, 2016.

16. Himsolt M. GML: A portable Graph File Format. 2010. URL: http://www.fim.uni-passau.de/fileadmin/files/lehrstuhl/brandenburg/projekte/gml/gml-technical-report.pdf.

17. Holt R.C., Schürr A., Sim S.E., Winter A. GXL: A graph-based standard exchange format for reengineering. Science of Computer Programming. 2006. Vol. 60, no. 2. P. 149–170. DOI: 10.1016/j.scico.2005.10.003.

18. Batagelj V., Mrvar A. Towards NetML Networks Markup Language. International Social Network Conference, London. 1995.

19. Kumar A.R.A., Rao S.V., Goswami D. NS3 Simulator for a Study of Data Center Networks. IEEE 12th International Symposium on Parallel and Distributed Computing, ISPDC 2013, Bucharest, Romania, June 27-30, 2013. IEEE, 2013. P. 224–231. DOI: 10.1109/ISPDC.2013.37.

20. Choudhury N., Mehta Y., Wilmarth T.L., *et al.* Scaling an Optimistic Parallel Simulation of Large-Scale Interconnection Networks. Proceedings of the 37th Winter Simulation Conference, Orlando, FL, USA, December 4-7, 2005. IEEE, 2005. P. 591–600. DOI: 10.1109/WSC.2005.1574299.

21. Lantz B., Heller B., McKeown N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. Proceedings of the 9th ACM Workshop on Hot Topics in Networks. HotNets 2010, Monterey, CA, USA, October 20-21, 2010. ACM, 2010. Article 19. DOI: 10.1145/1868447.1868466.

22. Navaridas J., Pascual J.A., Erickson A., *et al.* INRFlow: An interconnection networks research flow-level simulation framework. Journal of Parallel and Distributed Computing. 2019. Vol. 130. P. 140–152. DOI: 10.1016/j.jpdc.2019.03.013.

23. Varga A. The OMNET++ discrete event simulation system. Proc. ESM'2001. 2001. Vol. 9.

24. Varga A., Hornig R. An Overview of the OMNeT++ Simulation Environment. Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops, SimuTools 2008, Marseille, France, March 3-7, 2008. ICST/ACM, 2008. Article 60. DOI: 10.5555/1416222.1416290.

25. Sabidussi G. Graph multiplication. Mathematische Zeitschrift. 1959. Vol. 72, no. 1. P. 446–457. DOI: 10.1007/bf01162967.

26. Weichsel P. The Kronecker Product of Graphs. Proceedings of The American Mathematical Society - PROC AMER MATH SOC. 1962. Vol. 13. DOI: 10.2307/2033769.

27. Geller D., Stahl S. The chromatic number and other functions of the lexicographic product. Journal of Combinatorial Theory, Series B. 1975. Vol. 19, no. 1. P. 87–95. DOI: 10.1016/0095-8956(75)90076-3.

28. Godsil C., McKay B. A new graph product and its spectrum. Bulletin of The Australian Mathematical Society – BULL AUST MATH SOC. 1978. Vol. 18. DOI: 10.1017/S0004972700007760.

29. Knuth D.E. The Art of Computer Programming: Volume 4A: Combinatorial Algorithms: Part 1. Addison-Wesley Professional, 2014. P. 27–28.

30. Chen D., Eisley N.A., Heidelberger P., *et al.* The IBM Blue Gene/Q Interconnection Network and Message Unit. Conference on High Performance Computing Networking, Storage and Analysis, SC 2011, Seattle, WA, USA, November 12-18, 2011. ACM, 2011. Article 26. DOI: 10.1145/2063384.2063419.

31. Bray T. The JavaScript Object Notation (JSON) Data Interchange Format: RFC / RFC Editor. 2017. No. 8259. URL: https://www.rfc-editor.org/rfc/rfc8259.txt.

32. IEEE Standard for Information Technology–Portable Operating System Interface (POSIX(R)) Base Specifications, Issue 7. IEEE Std 1003.1-2017 (Revision of IEEE Std 1003.1-2008). 2018. P. 1–3951. DOI: 10.1109/IEEESTD.2018.8277153.

33. Stunkel C.B., Graham R.L., Shainer G., *et al.* The high-speed networks of the Summit and Sierra supercomputers. IBM J. Res. Dev. 2020. Vol. 64, no. 3/4. 3:1–3:10. DOI: 10.1147/JRD.2020.2967330.

34. Liao X.-K., Pang Z.-B., Wang K.-F., *et al.* High Performance Interconnect Network for Tianhe System. Journal of Computer Science and Technology. 2015. Vol. 30, no. 2. P. 259–272. DOI: 10.1007/s11390-015-1520-7.

35. Takizawa S. AI Bridging Cloud Infrastructure (ABCI) and its Communication Performance. 2019. URL: http://mug.mvapich.cse.ohio-state.edu/mug/19/ 7th Annual MVAPICH User Group Meeting.

36. Voevodin V., Antonov A., Nikitenko D., *et al.* Supercomputer Lomonosov-2: Large Scale, Deep Monitoring and Fine Analytics for the User Community. Supercomputing Frontiers and Innovations. 2019. Vol. 6, no. 2. P. 4–11. DOI: 10.14529/jsfi190201.

37. PARALLEL.RU. Supercomputer “Lomonosov-2” | PARALLEL.RU – Information and analytical center for parallel computing on the internet. URL: https://parallel.ru/cluster/lomonosov2.html (accessed: 15.05.2020).

УДК 004.43, 004.2, 004.451 DOI: 10.14529/cmse220303

УЧЕБНЫЙ КУРС «ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МОДЕЛИ ONEAPI»*

© 2022 А.В. Сысоев, А.В. Горшков, В.Д. Волокитин, Н.В. Шестакова, И.Б. Мееров

Нижегородский государственный университет им. Н.И. Лобачевского (603022 Нижний Новгород, пр. Гагарина, д. 23)

E-mail: sysoyev@vmk.unn.ru, anton.v.gorshkov@gmail.com, volokitin@itmm.unn.ru, shestakova@vmk.unn.ru, meerov@vmk.unn.ru

Поступила в редакцию: 15.09.2022

Современные высокопроизводительные вычислительные системы в массе своей являются гетерогенными. Разработка параллельных программ, способных использовать весь потенциал таких систем, сопряжена со значительными сложностями — требуется не только применять соответствующие языки и технологии программирования, но и учитывать особенности центральных и графических процессоров, влияющие в том числе на схемы организации параллелизма и работу с памятью. На упрощение процесса разработки таких программ направлена модель гетерогенного программирования oneAPI, представленная компанией Intel, и ее ключевой компонент — язык Data Parallel C++, позволяющий разрабатывать переносимые высокопроизводительные программы для CPU, GPU, FPGA и других устройств. В статье представлен учебный курс по oneAPI, разработанный в ННГУ им. Н. И. Лобачевского. Курс направлен на изучение широкого спектра вопросов, связанных с высокопроизводительными вычислениями с использованием моделей, методов и инструментов параллельного программирования на платформах Intel. В статье представлена концепция курса, описана его структура, категории слушателей, которым он может быть интересен, и варианты построения курса в зависимости от уровня подготовки аудитории.

Ключевые слова: образование, высокопроизводительные вычисления, параллельное программирование, гетерогенные вычислительные системы, Data Parallel C++, SYCL.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Сысоев А.В., Горшков А.В., Волокитин В.Д., Шестакова Н.В., Мееров И.Б. Учебный курс «Программирование с использованием модели oneAPI» // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 45–58. DOI: 10.14529/cmse220303.

Введение

Одна из тенденций последнего десятилетия заключается в появлении широкого спектра вычислительных архитектур, уникальные особенности которых требуют разработки, освоения и применения соответствующих языков и средств программирования. Известно, что затраты на подготовку специалистов, способных квалифицированно использовать языки и средства программирования для создания высокопроизводительных программ, существенно отличаются. Так, например, языки C, C++ и Fortran, широко применяемые для разработки программных средств численного моделирования, изучаются при подготовке бакалавров по всему миру, тогда как CUDA, позволяющая программировать для GPU, требует существенно больших трудозатрат (и предварительных знаний) при ее освоении. Принимая во внимание тот факт, что основные средства программирования для CPU и GPU не совместимы, а особенности разных вычислительных архитектур требуют использо-

*Статья рекомендована к публикации программным комитетом Международной конференции «Суперкомпьютерные дни в России – 2022».

вания специфичных приемов оптимизации кода, нередко требуется разработка, поддержка и развитие нескольких версий программного кода. На это обычно не хватает ресурсов.

Для решения этой проблемы разрабатываются универсальные средства разработки программ (языки, расширения для языков, библиотеки), которые позволяют использовать один язык параллельного программирования для разных вычислительных устройств. Среди таких средств можно отметить OpenCL, OpenACC, SYCL, alpaka, KOKKOS и другие разработки. В данной работе мы основываемся на недавно представленном языке гетерогенного программирования Data Parallel C++ (DPC++) [1], который основан на стандарте языка SYCL и является частью модели oneAPI, вобравшей в себя практически весь набор библиотек и инструментов программирования, разработанный в компании Intel за последние десятилетия. Наличие обширного и хорошо проработанного инструментария (компиляторы, профилировщики, отладчик, анализатор кода, математические библиотеки, средства организации и поддержки параллелизма, проблемно-ориентированные библиотеки) и сообщества разработчиков и пользователей является достоинством oneAPI и DPC++, и в целом дает надежду на успешное развитие этого проекта. Несмотря на то, что вопрос переносимости производительности (performance portability) далек от своего решения, сам факт использования единого инструментария уже является существенным шагом вперед. Этими соображениями и обусловлен наш интерес к разработке учебного курса по модели oneAPI и языку DPC++.

Учебный курс по любой технологии параллельного программирования опирается на два предположения относительно слушателя, приступающего к его освоению.

1. Слушатель владеет базовыми знаниями по основам алгоритмизации, модульному программированию, алгоритмам и структурам данных, а также имеет опыт применения этих знаний, как минимум, на учебных задачах.
2. Слушатель владеет языком программирования, на котором построена рассматриваемая технология, как минимум, в той степени, при которой в представленных в курсе примерах программ ему не встречаются незнакомые конструкции языка.

Другими словами, в рамках курса по технологии параллельного программирования слушателя не нужно учить разработке последовательных программ на требуемом этой технологией языке. Однако в силу новизны языка DPC++ курс по модели oneAPI может быть интересен разным категориям слушателей с существенно разным уровнем подготовки на входе. Таким образом, в основу построения структуры курса была положена идея модульности и возможности адаптации его наполнения и длительности, исходя из целей обучения и особенностей аудитории.

Дальнейшее изложение в статье построено следующим образом. В разделе 1 представлен обзор положения дел в области подготовки специалистов по высокопроизводительным вычислениям. В разделе 2 изложены требования к уровню знаний слушателя курса, как обязательные, так и желательные, а также описаны категории слушателей, которым курс по oneAPI может быть полезен/интересен. В разделе 3 описана структура курса, в разделе 4 обобщен первый опыт его прочтения.

1. Аналогичные работы

Разработка курсов по параллельным вычислениям [2–6] ведется в России и в мире на протяжении многих лет. Среди российских разработок отметим курсы по параллельным базам данных [7, 8], по основам параллельных вычислений для систем с общей [9, 10] и с рас-

пределенной [10] памятью, по параллельным алгоритмам решения задач вычислительной математики [11], по математическим и алгоритмическим основам параллельных вычислений [12], по преподаванию элементов параллельного программирования в школе [13]. Значительный вклад в развитие образовательного сообщества в России внес масштабный проект «Суперкомпьютерное образование», объединивший усилия многих специалистов по всей стране, на протяжении нескольких десятилетий развивавших данную тематику ([14–17]), а также создание Суперкомпьютерного консорциума университетов России.

Несмотря на обилие высококачественных образовательных материалов по параллельным вычислениям, появляющиеся новые технологии требуют разработки новых курсов. В этой связи необходимо особо отметить область гетерогенных вычислений. Давно установлено, что само по себе параллельное программирование сложнее «обычного» программирования, не только в плане вопросов организации данных и алгоритмов, а также производительности вычислений, но и, особенно, в плане тестирования и отладки кода. Не случайно ведущие производители программного обеспечения направляют значительные усилия на разработку соответствующего инструментария (например, отладчика Intel Inspector, входящего в пакет Parallel Studio). Практика показала, что подготовка специалистов по GPU-программированию — еще более сложный вопрос из-за повышенного «порога вхождения» в тему. В сравнении с параллельным программированием для CPU, в данном случае повышаются требования к слушателям в части изучения архитектуры графических процессоров и соответствующих подходов к оптимизации производительности, отличающихся от аналогичных методов для обычных процессоров. Дополнительно усложняется и отладка. Описанные сложности являются одним из дополнительных факторов при разработке моделей, методов и языков для гетерогенного программирования, цель которых — предоставить единые средства для разработки программ, которые могут работать на разных архитектурах, а при необходимости использовать в расчетах разнородные устройства. Одним из таких языков является SYCL и его расширение от компании Intel — Data Parallel C++ [1]. Именно этому языку и, в целом, модели программирования OneAPI и посвящен данный курс.

К настоящему времени учебных материалов по SYCL и DPC++ разработано не так много, соответствующее развитие образовательной экосистемы только начинается. В этой связи прежде всего отметим книгу [1], недавно опубликованные курсы [18, 19], а также серию лекций от отдельных авторов [20] и коммерческих компаний [21]. В целом появление подобных материалов наряду с организацией конференций и воркшопов показывают наличие интереса к данной тематике и доказывают актуальность разработки новых учебных курсов.

2. Слушатели курса и требования к уровню их подготовки

Модель oneAPI представляет язык разработки гетерогенных параллельных программ DPC++. Язык DPC++ основан на стандарте языка программирования SYCL, который, в свою очередь, в значительный степени является расширением языка программирования C++ [22–24] (стандарта 11 и выше). Таким образом, курс по oneAPI добавляет к базовым требованиям к слушателю — знаниям по основам алгоритмизации и структурам данных — еще и владение языком программирования C++, а также объектно-ориентированным программированием (ООП). Как ООП, так и базовый C++ не могут быть с хоть сколько-нибудь необходимой детальностью изложены в рамках курса по oneAPI и являются обязательными входными требованиями.

При этом, несмотря на то, что существенное обновление языка, привнесенное стандартом C++ 11 и, пусть и в меньшей степени, продолженное последующими стандартами (14, 17 и 20), произошло уже более десяти лет назад, многие учебные курсы, использующие язык C++ в качестве базового при изложении концепций и технологий программирования, включая ООП, все еще ориентированы на изложение C++ 98/03. Как следствие, те конструкции DPC++, которые основаны на возможностях современного C++, могут быть слушателям, с ними незнакомыми, не вполне понятны. Выходов из ситуации видится два: 1) включить во входные требования к курсу по oneAPI знания по C++ стандартов 11 и выше; 2) изложить используемые в DPC++ конструкции современного C++ непосредственно в курсе по oneAPI.

Проанализировав совместно с коллегами из компании Intel язык DPC++ с точки зрения набора используемых им элементов C++ из стандартов, начиная с 11-го, мы пришли к выводу, что этот набор ограничен в достаточной степени, чтобы в рамках курса по oneAPI можно было реализовать именно второй вариант, не повышая тем самым входные требования к потенциальным слушателям.

Аналогичным образом обстоит дело еще с двумя областями знаний: архитектуры вычислительных систем и операционные системы. Качественное освоение любого курса по параллельному программированию и высокопроизводительным вычислениям невозможно без определенного уровня знаний об архитектурах параллельных вычислительных систем [25, 26] и ключевых возможностях современных операционных систем [27, 28], на которых базируется разработка параллельных программ. Предварительное изучение соответствующих курсов безусловно будет полезно перед тем, как приступать к знакомству с миром разработки параллельных программ, но, тем не менее, не является строго обязательным. Необходимый минимум сведений из архитектур и операционных систем может быть изложен непосредственно в курсе по параллельному программированию в виде дополнительных начальных модулей, которые для подготовленных слушателей могут быть как сокращены, так и опущены вовсе.

Как уже было отмечено во введении, курс по oneAPI может быть интересен разным категориям слушателей. В первую очередь, это студенты бакалавриата, уже прошедшие через курсы по основам программирования, алгоритмам и структурам данных, и освоившие язык C++ и технологию ООП. Курс может быть полезен магистрантам, возможно уже знакомым с какими-то из технологий разработки параллельных программ (OpenMP, TBB, MPI и др.). Курс может заинтересовать и преподавателей вузов, читающих учебные курсы по технологиям параллельного программирования и желающих их актуализировать. Наконец, в силу новизны языка DPC++ курс может привлечь и специалистов в области высокопроизводительных вычислений и суперкомпьютерных технологий, желающих попробовать и/или освоить новую технологию. Очевидно, что каждой из указанных категорий слушателей требуется разная степень детальности изложения материала, каждая из них на входе в курс обладает разными входными знаниями и опытом. Именно по этой причине структура курса по oneAPI построена в виде отдельных модулей, часть из которых может быть опущена и/или сокращена при соответствующем уровне подготовки слушателей.

3. Структура курса

Курс состоит из 6 модулей:

1. Введение в мир суперкомпьютерных вычислений.

2. Архитектурные механизмы, влияющие на производительность. Уровни параллелизма.
3. Операционные системы. Аспекты параллелизма.
4. Программирование на языке Data Parallel C++.
5. Программные библиотеки и инструменты oneAPI.
6. Высокопроизводительные вычисления и научное моделирование. Примеры использования.

Модули 2 и 3 относятся к необязательным и, как уже было сказано выше, могут быть сокращены или опущены.

Модуль по DPC++ является основным в курсе. Именно к нему, в первую очередь, относятся сформулированные в разделе 2 входные требования к слушателям.

Модуль 5 содержит обзорные материалы по библиотекам, включенным компанией Intel в oneAPI, а также инструментам, поддерживающим процесс разработки параллельных программ.

В модуле 6 демонстрируется использование библиотек и инструментов oneAPI при решении задач в научных проектах.

Рассмотрим содержание модулей более подробно.

3.1. Модуль «Введение в мир суперкомпьютерных вычислений»

Продолжительность модуля — одна лекция.

Предварительные требования к освоению модуля отсутствуют.

В модуле дается обзор предметной области суперкомпьютерных вычислений, показывается разнообразие типов вычислительных систем и подходов к разработке параллельных программ. Обсуждаются сильные и слабые стороны подходов. Демонстрируется, чем DPC++ может помочь разработчикам параллельных программ для гетерогенных вычислительных систем.

3.2. Модуль «Архитектурные механизмы, влияющие на производительность. Уровни параллелизма»

Продолжительность модуля — две лекции.

Предварительные требования к освоению модуля отсутствуют.

В модуле рассматриваются ключевые особенности современных вычислительных архитектур, с точки зрения высокопроизводительных вычислений и суперкомпьютерных технологий. Обсуждается их влияние на производительность и эффективность параллельных программ.

3.3. Модуль «Операционные системы. Аспекты параллелизма»

Продолжительность модуля — две лекции, одна практика.

Предварительные требования к освоению модуля отсутствуют.

В модуле дается обзор ключевых особенностей современных операционных систем с точки зрения разработки параллельных программ. Рассматриваются процессы, потоки и планирование времени центрального процессора. Обсуждаются вопросы синхронизации выполнения потоков.

3.4. Модуль «Программирование на языке Data Parallel C++»

Модуль состоит из пяти лекций и пяти практик.

Предварительные требования к освоению модуля: знание технологий структурного, модульного и объектно-ориентированного программирования. Знание C++ 98/03 или C++ 11. Представление об архитектуре многоядерных центральных процессоров. Представление об архитектуре графических процессоров. Представление о потоках и их синхронизации.

Поскольку данный модуль является в курсе основным, его структуру и наполнение рассмотрим более детально.

Лекция 0. Элементы современного C++

Рассматриваются конструкции языка C++, появившиеся в нем в стандарте C++ 11 и последующих, активно используемые при разработке параллельных программ на языке DPC++.

Практика 0. Элементы современного C++ 11, используемые в DPC++

Лекция 1. Введение в Data Parallel C++

Основные темы лекции:
- обзор Intel oneAPI — концепция, языки, библиотеки, инструменты;
- базовые понятия гетерогенного программирования — хост, устройство, ядро, очередь, типы памяти;
- язык DPC++, основные особенности — язык высокого уровня, единый исходный код для всех поддерживаемых устройств, стандартный C++;
- модель SPMD (single program, multiple data);
- обработка ошибок;
- вывод информации с устройства.

Практика 1. Hello World

Задачи практики:
- вывести в консоль все доступные платформы и устройства средствами DPC++;
- разработать ядро (kernel), чтобы напечатать “Hello” из ядра каждого из устройств.

Лекция 2. Исполнение ядер в Data Parallel C++

Основные темы лекции:
- модель исполнения ядра DPC++;
- запуск ядра — анатомия ядра, типы параллелизма, индексация потоков/групп;
- измерение времени выполнения ядра;
- групповые алгоритмы.

Практика 2. Вычисление двойного интеграла методом сумм Римана

Задачи практики:
- реализовать на DPC++ вычисление заданного интеграла с использованием средней суммы Римана;
- сравнить производительность ядра для разных платформ и числа разбиений.

Лекция 3. Управление памятью в Data Parallel C++

Основные темы лекции:
- модель памяти;
- буферы и объекты доступа (аксессоры);
- типы аксессоров;
- атомарные операции;
- унифицированная разделяемая память;
- пример: матрично-векторное умножение.

Практика 3. Решение систем линейных уравнений методом Якоби

Задачи практики:
- реализовать на DPC++ решение системы уравнений методом Якоби;
- разработать три версии — с использованием аксессоров, разделяемой памяти (модель USM), памяти устройства;
- сравнить производительность для разных версий, устройств и входных данных.

Лекция 4. Оптимизация программ на Data Parallel C++

Основные темы лекции:
- базовые рекомендации;
- глобальная память;
- локальная память;
- подгруппы;
- приватная память;
- векторизация;
- высокопроизводительные библиотеки.

Практика 4. Умножение матриц

Задачи практики:
- реализовать на DPC++ умножение матриц;
- разработать четыре версии — «наивную», блочную, векторизованную блочную, с использованием oneMKL;
- сравнить производительность для разных версий, устройств и входных данных.

3.5. Модуль «Программные библиотеки и инструменты oneAPI»

Продолжительность модуля — две лекции, одна практика, три мастер-класса.

Предварительные требования к освоению модуля отсутствуют.

В модуле дается краткий обзор библиотек IPP, oneVPL, oneDAL, oneMKL, oneTBB, oneDPL с точки зрения их использования при разработке параллельных программ. Более детально рассматривается работа с библиотекой oneVPL. В виде мастер-классов демонстрируется оптимизация программ с использованием Intel VTune и Intel Advisor, а также вопросы высокопроизводительного вывода нейронных сетей с Intel Distribution of OpenVINO toolkit.

3.6. Модуль «Высокопроизводительные вычисления и научное моделирование. Примеры использования»

Продолжительность модуля — четыре мастер-класса.

Предварительные требования к освоению модуля: знание технологий структурного, модульного и объектно-ориентированного программирования; знание C++ 11; знание языка DPC++.

В модуле предполагается продемонстрировать использование библиотек и инструментов oneAPI при решении задач в научных проектах. Запланированы следующие мастер-классы:

- портирование модуля движения частиц [29–31] на DPC++ (анализ и оптимизация производительности на CPU и GPU с использованием Intel VTune);
- анализ финансовых рынков: вычисление формулы Блэка—Шоулса (анализ и оптимизация производительности на CPU и GPU с использованием Intel VTune);
- интегрирование уравнений Максвелла методом FDTD (анализ и оптимизация производительности на CPU и GPU);
- вычисления в смешанной точности на CPU и GPU на примере задачи интегрирования уравнений Максвелла методом FDTD.

В настоящий момент готовы первый и второй мастер-класс, остальные находятся в разработке.

4. Внедрение курса в учебный процесс

При разработке курса мы ориентировались на его всестороннюю апробацию, сбор отзывов от разных категорий слушателей и соответствующую адаптацию материалов. В начале мы провели серию открытых лекций, преимущественно ориентированных на студентов 3-6 курсов, которые ранее уже занимались проблематикой высокопроизводительных вычислений. На основе полученной обратной связи, а также в ходе консультаций с разработчиками oneAPI была сформирована программа курса, описанная в предыдущем разделе статьи. После разработки основной части материалов курс был прочитан студентам магистерской программы «Вычислительные методы и суперкомпьютерные технологии» [32]. С использованием материалов курса был проведен мастер-класс на конференции «Суперкомпьютерные дни в России» (Москва, сентябрь 2021 года). Материалы курса прошли рецензирование у разработчиков oneAPI. По итогам была разработана и опубликована текущая версия материалов курса на русском [33] и английском [34] языках.

В январе 2022 года была организована в онлайн-формате программа повышения квалификации по программированию с использованием модели oneAPI для преподавателей вузов и сотрудников НИИ России. Получены положительные отзывы по результатам прохождения обучения, материалы переданы коллегам для дальнейшего использования.

5. Заключение

В статье представлен учебный курс по oneAPI, разработанный в ННГУ им. Н. И. Лобачевского. Приведена модульная структура курса и указаны соображения, положенные в основу его организации. Описаны категории слушателей, которым курс по oneAPI может быть полезен/интересен, а также изложены требования к предварительному уровню знаний слушателей.

Отличительная особенность курса — демонстрация использования возможностей oneAPI при решении практических задач из разных предметных областей. Часть запланированных примеров уже включена в курс, еще несколько из реальных научных проектов Центра суперкомпьютерных технологий ННГУ находятся в процессе доработки.

Учебный курс разработан в Центре компетенций oneAPI ННГУ при поддержке компании Intel. Работа частично поддержана Министерством науки и высшего образования, проект № 0729-2020-0055.

Литература

1. Reinders J., Ashbaugh B., Brodman J., *et al.* Data Parallel C++ Mastering: DPC++ for Programming of Heterogeneous Systems using C++ and SYCL. Apress Berkeley, CA, 2021. 548 p. DOI: 10.1007/978-1-4842-5574-2.
2. Воеводин В.В. Параллельные вычисления и математическое образование // Математика в высшем образовании. 2005. Т. 3. С. 9–26. URL: https://www.mathedu.ru/text/mvo_2005_3/p9/.
3. Nevison C.H. Parallel computing in the undergraduate curriculum // Computer. 1995. Vol. 28, no. 12. P. 51–56. DOI: 10.1109/2.476199.
4. El-Rewini H., Lewis T.G. Distributed and parallel computing. Manning Publications Co., 1998.
5. Crichlow J.M. An introduction to distributed and parallel computing. Prentice-Hall, 1988.
6. Grama A., Gupta A., Karypis G., Kumar V. Introduction to parallel computing. Addison-Wesley, 2003. 664 p.
7. Цымблер М.Л., Соколинский Л.Б., Лепихов А.В. Прототипирование параллельной СУБД как основа учебного курса по параллельным системам баз данных // Суперкомпьютерные системы и их применение. 2004. С. 212–217.
8. Соколинский Л.Б., Цымблер М.Л. Лекции по курсу «Параллельные системы баз данных». URL: https://pdbs.susu.ru/CourseManual.html (дата обращения: 14.09.2022).
9. Биллиг В.А. Параллельные вычисления и многопоточное программирование. Москва: НОУ «ИНТУИТ», 2016.
10. Гергель В.П. Высокопроизводительные вычисления для многопроцессорных многоядерных систем. Москва: Издательство Московского университета, 2010. 534 с.
11. Баркалов К.А., Мееров И.Б., Бастраков С.И. Об опыте разработки и чтения курса лекций «Параллельные численные методы» // Суперкомпьютерные дни в России: Труды международной конференции, 28–29 сентября, 2015. Москва: Московский государственный университет, 2015. С. 772–775.
12. Антонов А.С., Воеводин В.В., Попова Н.Н. Параллельная структура алгоритмов и подготовка специалистов по вычислительным технологиям // Актуальные проблемы прикладной математики, информатики и механики. 2017. С. 4–11.
13. Плаксин М.А. О пропедевтике параллельных вычислений в школьной информатике // Информатика и образование. 2016. № 10. С. 27–36.

14. Антонов А.С., Воеводин В.В., Гергель В.П., Соколинский Л.Б. Системный подход к суперкомпьютерному образованию // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. Т. 2, № 2. С. 5–17. DOI: 10.14529/cmse130201.

15. Воеводин В.В., Гергель В.П., Соколинский Л.Б., др. Развитие системы суперкомпьютерного образования в России: текущие результаты и перспективы // Вестник Нижегородского университета. 2013. Т. 2, № 2. С. 203–209.

16. Воеводин В.В., Гергель В.П. Суперкомпьютерное образование: третья составляющая суперкомпьютерных технологий // Вычислительные методы и программирование. 2010. Т. 11. С. 117–122.

17. Гергель В.П., Демкин В.П., Коньков К.А. и др. Проект «Суперкомпьютерное образование»: 2012 год // Вестник Нижегородского университета. 2013. 1(1). С. 12–16.

18. Fuentes J., López D., González S. Teaching Heterogeneous Computing Using DPC++ // IEEE International Parallel and Distributed Processing Symposium, IPDPS Workshops 2022, Lyon, France, May 30 - June 3, 2022. IEEE, 2022. P. 354–360. DOI: 10.1109/ipdpsw55747.2022.00069.

19. Heterogeneous programming with SYCL. URL: https://enccs.github.io/sycl-workshop/ (дата обращения: 14.09.2022).

20. Владимиров К. Краткий обзор современного гетерогенного программирования в OpenCL и SYCL а также расширения от Intel, составляющие DPC++. URL: https://www.youtube.com/watch?v=fAVOPJLu2qA (дата обращения: 14.09.2022).

21. SYCL Training. URL: https://www.codeplay.com/solutions/sycl-training/ (дата обращения: 14.09.2022).

22. Stroustrup B. The C++ Programming Language (4th Edition). Addison-Wesley, 2013. 1376 p.

23. Josuttis N.M. The C++ Standard Library – A Tutorial and Reference, second edition. Addison-Wesley, 2012. 1136 p.

24. C++ FAQ. URL: https://isocpp.org/wiki/faq (дата обращения: 14.09.2022).

25. Паттерсон Д., Хеннесси Д. Архитектура компьютера и проектирование компьютерных систем. Издательский Дом ПИТЕР, 2012. 784 с.

26. Таненбаум Э. Архитектура компьютера. 4-е изд. Издательский Дом ПИТЕР, 2015. 704 с.

27. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. Издательский Дом ПИТЕР, 2015. 1120 с.

28. Коньков К., Карпов В. Основы операционных систем: курс лекций: учеб. пособие для вузов. НОУ «ИНТУИТ», 2016.

29. Код Hi-Chi. URL: https://github.com/hi-chi/pyHiChi (дата обращения: 14.09.2022).

30. О проекте. URL: http://hpc-education.unn.ru/en/research/overview/laser-plasma (дата обращения: 14.09.2022).

31. Volokitin V., Bashinov A.V., Efimenko E., *et al.* High Performance Implementation of Boris Particle Pusher on DPC++. A First Look at oneAPI // Parallel Computing Technologies - 16th International Conference, PaCT 2021, Kaliningrad, Russia, September 13-18, 2021, Proceedings. Vol. 12942 / ed. by V. Malyshkin. Springer, 2021. P. 288–300. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-86359-3_22.

32. Meyerov I.B., Sysoyev A., Pirova A., *et al.* Bridging the Gap Between Applications and Supercomputing: A New Master's Program in Computational Science // Supercomputing - 5th Russian Supercomputing Days, RuSCDays 2019, Moscow, Russia, September 23-24, 2019, Revised Selected Papers. Vol. 1129 / ed. by V.V. Voevodin, S. Sobolev. Springer, 2019. P. 529–541. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_43.

33. Программирование с использованием модели oneAPI. URL: https://hpc-education.unn.ru/ru/центр-компетенций-oneapi-в-ннгу/курс-oneapi (дата обращения: 14.09.2022).

34. Programming with oneAPI. URL: https://hpc-education.unn.ru/en/the-oneapi-center-of-excellence/programming-with-oneapi (дата обращения: 14.09.2022).

Сысоев Александр Владимирович, к.т.н., доцент кафедры математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Горшков Антон Валерьевич, к.т.н., доцент кафедры математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Волокитин Валентин Дмитриевич, младший научный сотрудник, кафедра математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Шестакова Наталья Валерьевна, старший преподаватель кафедры математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

Мееров Иосиф Борисович, к.т.н., доцент, зам.зав. кафедрой математического обеспечения и суперкомпьютерных технологий, Нижегородский государственный университет им. Н.И. Лобачевского (национальный исследовательский университет) (Нижний Новгород, Российская Федерация)

DOI: 10.14529/cmse220303

PROGRAMMING WITH ONEAPI: A NEW COURSE ON HETEROGENEOUS COMPUTING

© 2022 A.V. Sysoyev, A.V. Gorshkov, V.D. Volokitin, N.V. Shestakova, I.B. Meyerov

National Research Lobachevsky State University of Nizhny Novgorod (pr. Gagarina 23, Nizhny Novgorod, 603022 Russia)

E-mail: sysoyev@vmk.unn.ru, anton.v.gorshkov@gmail.com, volokitin@itmm.unn.ru, shestakova@vmk.unn.ru, meerov@vmk.unn.ru

Received: 15.09.2022

Modern high-performance computing systems are mostly heterogeneous. The development of parallel programs that can use the full potential of such systems is fraught with significant difficulties. It is required not only to use the appropriate programming languages and technologies, but also to take into account the features of central and graphic processors that affect, among other things, the implementation of parallel schemes and memory management. The oneAPI heterogeneous programming model presented by Intel is aimed at simplifying the process of developing such programs, and its key component is the Data Parallel C ++ language, which allows developing portable high-performance programs for CPU, GPU, FPGA and other devices. The article presents a training course on oneAPI, developed at the Lobachevsky University. The course is aimed at studying a wide range of issues related to high-performance computing using models, methods and tools for parallel programming on Intel platforms. The article presents the concept of the course, describes its structure, categories of listeners who may be interested in it, and options for building a course depending on the level of preparation of the audience.

Keywords: education, high performance computing, parallel programming, heterogeneous computing systems, Data Parallel C++, SYCL.

FOR CITATION

Sysoyev A.V., Gorshkov A.V., Volokitin V.D., Shestakova N.V., Meyerov I.B. Programming with oneAPI: New Course Heterogenious Computing. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 3. P. 45–58. (in Russian) DOI: 10.14529/cmse220303.

References

1. Reinders J., Ashbaugh B., Brodman J., *et al.* Data Parallel C++ Mastering: DPC++ for Programming of Heterogeneous Systems using C++ and SYCL. Apress Berkeley, CA, 2021. 548 p. DOI: 10.1007/978-1-4842-5574-2.
2. Voevodin V.V. Parallel Computing and Mathematics Education. Mathematics in higher education. 2005. Vol. 3. P. 9–26. URL: https://www.mathedu.ru/text/mvo_2005_3/p9/ (in Russian).
3. Nevison C.H. Parallel computing in the undergraduate curriculum. Computer. 1995. Vol. 28, no. 12. P. 51–56. DOI: 10.1109/2.476199.
4. El-Rewini H., Lewis T.G. Distributed and parallel computing. Manning Publications Co., 1998.

5. Crichlow J.M. An introduction to distributed and parallel computing. Prentice-Hall, 1988.
6. Grama A., Gupta A., Karypis G., Kumar V. Introduction to parallel computing. Addison-Wesley, 2003. 664 p.
7. Zymbler M.L., Sokolinsky L.B., Lepihov A.V. Prototyping a Parallel DBMS as the Basis for a Training Course on Parallel Database Systems. Supercomputer systems and their applications. 2004. P. 212–217. (in Russian).
8. Sokolinsky L.B., Zymbler M.L. Training course "Parallel database systems". URL: `https://pdbs.susu.ru/CourseManual.html` (accessed: 14.09.2022) (in Russian).
9. Billig V.A. Parallel Computing and Multithreaded Programming. Moscow: INTUIT, 2016. (in Russian).
10. Gergel V.P. High performance computing for multiprocessor multicore systems. Moscow: Moscow University Press, 2010. 534 p. (in Russian).
11. Barkalov K.A., Meyerov I.B., Bastrakov S.I. About the experience of developing and teaching the course of lectures "Parallel Numerical Methods". Russian Supercomputer Days in Russia: Proceedings of the international conference, September 28–29, 2015. Moscow: Moscow State University, 2015. P. 772–775. (in Russian).
12. Antonov A.S., Voevodin V.V., Popova N.N. Parallel structure of algorithms and training of specialists in computing technologies. Actual problems of applied mathematics, informatics and mechanics. 2017. P. 4–11. (in Russian).
13. Plaksin M.A. On propaedeutics of parallel computing in school informatics. Computer science and education. 2016. No. 10. P. 27–36. (in Russian).
14. Antonov A.S., Voevodin V.V., Gergel V.P., Sokolinsky L.B. A systematic approach to supercomputing education. Bulletin of the South Ural State University. Computational Mathematics and Software Engineering. 2013. Vol. 2, no. 2. P. 5–17. (in Russian) DOI: `10.14529/cmse130201`.
15. Voevodin V.V., Gergel V.P., Sokolinsky L.B., *et al.* Development of the system of supercomputer education in Russia: current results and prospects. Bulletin of the Nizhny Novgorod University. 2013. Vol. 2, no. 2. P. 203–209. (in Russian).
16. Voevodin V.V., Gergel V.P. Supercomputer education: the third component of supercomputer technologies. Computational methods and programming. 2010. Vol. 11. P. 117–122. (in Russian).
17. Gergel V.P., Demkin V.P., Konkov K.A., *et al.* Project "Supercomputer Education": 2012. Bulletin of the Nizhny Novgorod University. 2013. 1(1). P. 12–16. (in Russian).
18. Fuentes J., López D., González S. Teaching Heterogeneous Computing Using DPC++. IEEE International Parallel and Distributed Processing Symposium, IPDPS Workshops 2022, Lyon, France, May 30 - June 3, 2022. IEEE, 2022. P. 354–360. DOI: `10.1109/ipdpsw55747.2022.00069`.
19. Heterogeneous programming with SYCL. URL: `https://enccs.github.io/sycl-workshop/` (accessed: 14.09.2022).
20. Vladimirov K. A brief overview of modern heterogeneous programming in OpenCL and SYCL, as well as extensions from Intel that are part of DPC++. URL: `https://www.youtube.com/watch?v=fAVOPJLu2qA` (accessed: 14.09.2022) (in Russian).

21. SYCL Training. URL: https://www.codeplay.com/solutions/sycl-training/ (accessed: 14.09.2022).

22. Stroustrup B. The C++ Programming Language (4th Edition). Addison-Wesley, 2013. 1376 p.

23. Josuttis N.M. The C++ Standard Library – A Tutorial and Reference, second edition. Addison-Wesley, 2012. 1136 p.

24. C++ FAQ. URL: https://isocpp.org/wiki/faq (accessed: 14.09.2022).

25. Patterson D., Hennessy J. Computer Architecture: A Quantitative Approach. Morgan Kaufmann, 2011. 856 p.

26. Tanenbaum A. Structured Computer Organization. Pearson, 2012. 704 p.

27. Tanenbaum A., Bos H. Modern Operating Systems. Pearson, 2014. 1136 p.

28. Konkov K., Karpov V. Fundamentals of operating systems: a course of lectures: a textbook for universities. NOU INTUIT, 2016. (in Russian).

29. Hi-Chi. URL: https://github.com/hi-chi/pyHiChi (accessed: 14.09.2022).

30. About project. URL: http://hpc-education.unn.ru/en/research/overview/laser-plasma (accessed: 14.09.2022).

31. Volokitin V., Bashinov A.V., Efimenko E., *et al.* High Performance Implementation of Boris Particle Pusher on DPC++. A First Look at oneAPI. Parallel Computing Technologies - 16th International Conference, PaCT 2021, Kaliningrad, Russia, September 13-18, 2021, Proceedings. Vol. 12942 / ed. by V. Malyshkin. Springer, 2021. P. 288–300. Lecture Notes in Computer Science. DOI: 10.1007/978-3-030-86359-3_22.

32. Meyerov I.B., Sysoyev A., Pirova A., *et al.* Bridging the Gap Between Applications and Supercomputing: A New Master's Program in Computational Science. Supercomputing - 5th Russian Supercomputing Days, RuSCDays 2019, Moscow, Russia, September 23-24, 2019, Revised Selected Papers. Vol. 1129 / ed. by V.V. Voevodin, S. Sobolev. Springer, 2019. P. 529–541. Communications in Computer and Information Science. DOI: 10.1007/978-3-030-36592-9_43.

33. Programming with oneAPI. URL: https://hpc-education.unn.ru/ru/центр-компетенций-oneapi-в-ннгу/курс-oneapi (accessed: 14.09.2022) (in Russian).

34. Programming with oneAPI. URL: https://hpc-education.unn.ru/en/the-oneapi-center-of-excellence/programming-with-oneapi (accessed: 14.09.2022).

УДК 541.65/.654, 548.31, 004.67, 004.054 DOI: 10.14529/cmse220304

МАСШТАБИРУЕМОСТЬ КВАНТОВОХИМИЧЕСКИХ РАСЧЕТОВ КРИСТАЛЛИЧЕСКИХ МАТЕРИАЛОВ НА СУПЕРКОМПЬЮТЕРЕ «ТОРНАДО ЮУРГУ»

© 2022 И.Д. Юшина, Ю.В. Матвейчук, Е.В. Барташевич

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: iushinaid@susu.ru, matveichukyv@susu.ru, bartashevichev@susu.ru

Поступила в редакцию: 17.08.2022

В работе обсуждаются рекомендации по рациональному использованию вычислительных ресурсов суперкомпьютера «Торнадо ЮУрГУ» для решения квантовохимических расчетных задач с учетом периодических граничных условий в программе CRYSTAL17. Решение таких задач необходимо для моделирования структуры и свойств кристаллов и, углеродных материалов. Проанализированы данные о временных характеристиках квантовохимических расчетов материалов различного состава и структуры, как для простых систем, таких, как трехмерный силицированный графит, так и для двумерных поверхностей углеродных материалов с диглицидиловым эфиром бисфенола А в роли сорбированной молекулы. Определены различия в масштабируемости расчетов в зависимости от их типа, размера периодически повторяющегося фрагмента структуры (элементарной ячейки), ее симметрии и входных параметров многоэлектронной системы, моделируемой на основе теории функционала электронной плотности в различных приближениях. Установлены рекомендуемые оптимальные параметры для различных типов химических и материаловедческих задач, решаемых в целях разработки цифровых двойников материалов. Обнаружено, что для относительно больших систем критическим является существенное увеличение требующегося объема временных файлов с увеличением числа используемых узлов, что приводит к неоптимальному режиму расчетов и возможному сбою. Показано, что масштабируемость расчетов колебательных характеристик кристаллов существенно ниже, чем для расчетов, направленных на поиск наиболее энергетически выгодной структуры кристалла, независимо от числа атомов в элементарной ячейке вычисляемой структуры.

Ключевые слова: цифровой двойник материалов, квантовохимические расчеты, функциональные материалы, оптимизация вычислений, масштабируемость.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Юшина И.Д., Матвейчук Ю.В., Барташевич Е.В. Масштабируемость квантовохимических расчетов кристаллических материалов на суперкомпьютере «Торнадо ЮУрГУ» // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 59–68. DOI: 10.14529/cmse220304.

Введение

Структурные модели химических соединений и материалов представляют собой, как правило, сложные равновесные атомно-молекулярные системы, в которых, в зависимости от уровня модели, учитывается распределение электронной плотности, конкретизированы позиции атомов, установлены типы химических связей. Точная оценка количественных взаимосвязей «структура – свойство» тесно связана с проблемой накопления и хранения химической информации, которая в перспективе позволит разрабатывать и внедрять технологии, связанные с использованием цифровых двойников химических соединений и материалов, что относится к задачам, отвечающая вызовам нашего времени. Цифровые двойники позволяют моделировать внедрение новых материалов, выявлять экономические и экологические риски при использовании инноваций еще на этапе проектирования. Отметим, что

оптимальный выбор материала при проектировании продукта является важнейшим вопросом при реализации устойчивого производства [1].

На протяжении многих лет отечественными и зарубежными исследователями проводились квантовохимические расчеты структуры и свойств химических структур с различной периодичностью для задач катализа [2], материаловедения [3], оптики и биологической активности [4]. Одной из отличительных особенностей этого круга задач в периодическом приближении является их возрастающая потребность в объеме вычислительных мощностей и большие временные затраты, необходимые для получения достоверных результатов. Одним из передовых программных продуктов для решения квантовохимических задач является CRYSTAL17, разрабатываемый в университете г. Турина [4]. Методические аспекты использования этой программы хорошо освещены в отечественной литературе в работах Р.А. Эварестова [5]. Однако вопросы оптимизации расчетов в условиях ограниченных вычислительных и временных ресурсов обычно остаются за рамками научных исследований, в то время как такие оценки необходимы для результативного использования имеющихся ресурсов и достижения результатов в условиях временных ограничений.

Интуитивный анализ более чем десятилетнего опыта расчетов в программе CRYSTAL17 на суперкомпьютере «Торнадо ЮУрГУ» показывает, что пользователи, как правило, не используют все имеющиеся возможности суперкомпьютера для минимизации компьютерного времени расчетов. Так, например, при работе с большей частью квантово-химического программного обеспечения редко обращается внимание на возможности существенного сокращения используемого объема дисковой памяти и обмена с этой памятью, что является в подавляющем большинстве случаев лимитирующим фактором времени расчетов. Как правило, работа с оперативной памятью является более предпочтительной в смысле временных затрат для очень многих прикладных расчетов, где не используются большие объемы промежуточных данных. Из этого правила есть исключения, например, расчет колебательных характеристик молекул или кристаллов с большим числом атомов в элементарной ячейке, когда установленной оперативной памяти недостаточно, и без большого количества дисковой памяти и интенсивного обмена с ней не обойтись. Необходимость систематического анализа возможностей ускорения решения задач с использованием квантовохимического программного обеспечения не теряет своей актуальности. Одной из таких возможностей является корректный выбор количества узлов суперкомпьютера, используемых для каждой отправляемой на расчет задачи, в зависимости от типа задачи, необходимых параметров сходимости расчета и величины рассчитываемых объектов. Для выяснения информации, позволяющей осуществлять правильный выбор при отправке задач на расчет, нами поставлены следующие задачи:

• определить возможность масштабирования вычислительных задач разных типов: оптимизации периодических 1D, 2D и 3D структур, решение спектральной задачи для 3D структур в CRYSTAL17 [6];

• определить оптимальные количества узлов, необходимых для оптимальных временных затрат с учетом загрузки внутренней сети и дисковой памяти для химических объектов разных размеров.

1. Вычислительные эксперименты

В соответствии с этими задачами нами были выбраны следующие объекты исследования: 2D поверхность, состоящая из трехатомного слоя частично замещенного кремнием

графита (силицированный графит) (1); 3D кристалл силицированного графита (2); бисфенол А, сорбированный на графеновом листе (3); бисфенола А, сорбированный на углеродной нанотрубке (УНТ) (4); кристаллическая структура каркасного соединения COF-102 (5) [7]. Основное тестирование осуществлялось для поиска равновесной геометрии системы. С одной стороны, это обязательная процедура моделирования, с другой стороны, для такого типа задач имеются статистические данные по эффективности распараллеливания алгоритма вычислений, предоставленные разработчиками программы [4]. Параметры моделирования для каждого объекта представлены в табл. 1, где № АО обозначено число атомных орбиталей на ячейку, а число шагов соответствует числу циклов на пути к найденному минимуму энергии. Графические результаты изменения времени расчета представлены на рис. 1–4.

Таблица 1. Структурные параметры объектов исследования и параметры расчетов

Объект	**Функционал**	**№ АО**	**Число атомов**	**Симметрия**	**Число шагов**
1	HSE06	116	8	P63/mmc	59
2	HSE06	344	24	P1	61
3	PBE0	3830	289	P1	75
4	PBE0	3606	273	P1	91
5	B3LYP	3252	13	I-43d	41

Формирование набора одновременно выполняемых задач осуществлялось в динамическом режиме в зависимости от получаемых результатов в текущем времени. Для задач поиска равновесной геометрии всех объектов был осуществлен предварительный подбор метода и параметров расчета для нахождения оптимизированной структуры. В дальнейшем одна и та же задача была стартована на разном числе узлов с контролем объема временных файлов на этапе оптимизации геометрии и прохождения 5 шагов. Диапазон узлов на один расчет выбирался от 1 до 140. Задачи расчета колебательных спектров проводились на небольшом числе узлов, так как ранее в стандартном режиме работы суперкомпьютера была неоднократно замечена вероятность срывов параллельно выполняемых задач из-за перегрузки сети при данном типе расчетов. В данном случае был протестирован диапазон от 1 до 4 узлов на 1 расчет [8].

Таблица 2. Характеристики кластера Торнадо ЮУрГУ

Характеристика	**Значение**
Количество процессорных узлов	480
Тип процессора	Intel Xeon X5680 (Gulftown, 6 ядер по 3.33 ГГц)
Оперативная память	24 Гб (DDR3-1333)
Операционная система	Linux CentOS
Соединительная сеть	InfiniBand QDR (40 Гбит/с)

2. Результаты и их обсуждение

В соответствии с данными, представленными на рис. 1–4, видно, что объем временных файлов растет линейно в зависимости от числа узлов для одной и той же задачи, независимо от объекта вычисления. Объем файлов также зависит от количества атомов в системе: от десятков мегабайт для 3D силицированного графита с восемью атомами в ячейке, до 11 Гб

для 2D комплекса бисфенола А с графеном с 273 атомами в ячейке. Следует отметить, что среди выбранных для исследования химических соединений со структурой, имеющей различную размерность — 2D и 3D, можно выделить хорошо масштабируемые: 2D поверхность силицированного графита и плохо масшабируемые: кристалл COF-102. Для количественной оценки времени расчета в зависимости от числа используемых узлов (табл. 3) было проведено сравнение вычислений при увеличении числа узлов в 10 раз в зависимости от их исходного числа: 1 и 10 узлов, 2 и 20 и так далее. Из табл. 3 мы видим, что для плохо масштабируемых вычислений показатель ускорения едва превышает единицу для любого соотношения начального и конечного числа узлов.

Таблица 3. Показатели ускорения расчета при использовании различного количества узлов и различных объектов 1–5

Отношение числа узлов	**1**	**2**	**3**	**4**	**5**
1/10	6.3	8.7	1.7	1.7	1.5
2/20	3.1	7.9	1.3	1.5	1.2
4/40	2.8	6.9	1.1	1.2	1.0
10/100	2.0	4.8	–	–	–
1/2	1.8	1.95	1.3	1.1	1.3

Однако для силицированного графита, как для 2D, так и 3D размерностей структуры, наблюдается существенный выигрыш во времени, хотя он значительно снижается при увеличении исходного числа узлов. Кроме того, для 2D варианта было протестировано самое большое число узлов на одну задачу — 140. Даже в этом случае наблюдалось небольшое сокращение времени расчета, хотя и несоизмеримое с увеличением используемых вычислительных мощностей (рис. 2). Для остальных рассмотренных объектов (рис. 1, 3, 4) выигрыш во времени расчета при увеличении вычислительных мощностей неоправданно мал. При этом очень быстро достигается пороговое значение числа узлов, когда дальнейшее увеличение этого числа вызывает рост времени расчета.

Таблица 4. Результаты расчета колебательных характеристик кристаллов силицированного графита (1) и COF-102 (5)

Параметр	**1**	**1**	**1**	**1**	**5**	**5**	**5**
Число узлов	1	2	4	8	1	2	4
Время расчета, мин	130	79	61	43	362	439	311
Объем временных файлов, Мб	213	245	289	326	12700	18000	19800

Оценка оптимального режима вычислений колебательных характеристик является более сложной задачей. В силу особенностей алгоритма такой тип задачи включает вычисление изменения энергии и вторых частных производных для элементов матрицы расстояний при смещении каждого симметрически-независимого атома по всем трем направлениям. В связи с этим решение спектральной задачи требует намного больших временных затрат. Тестирование решения спектральной задачи осуществлялось на кристаллических высокосимметричных объектах. В соответствии с полученными данными отметим, что для кристаллов с небольшим числом атомов в ячейке (силицированный графит 2D) и для крупной

ячейки кристаллического каркаса COF-102 наблюдалась в целом аналогичная картина, что и в задаче поиска равновесной геометрии. Однако если для COF-102 оптимизация геометрии являлась практически не масштабируемым процессом, то для расчета колебательных характеристик выигрыш во времени при увеличении числа узлов оказывается более существенным.

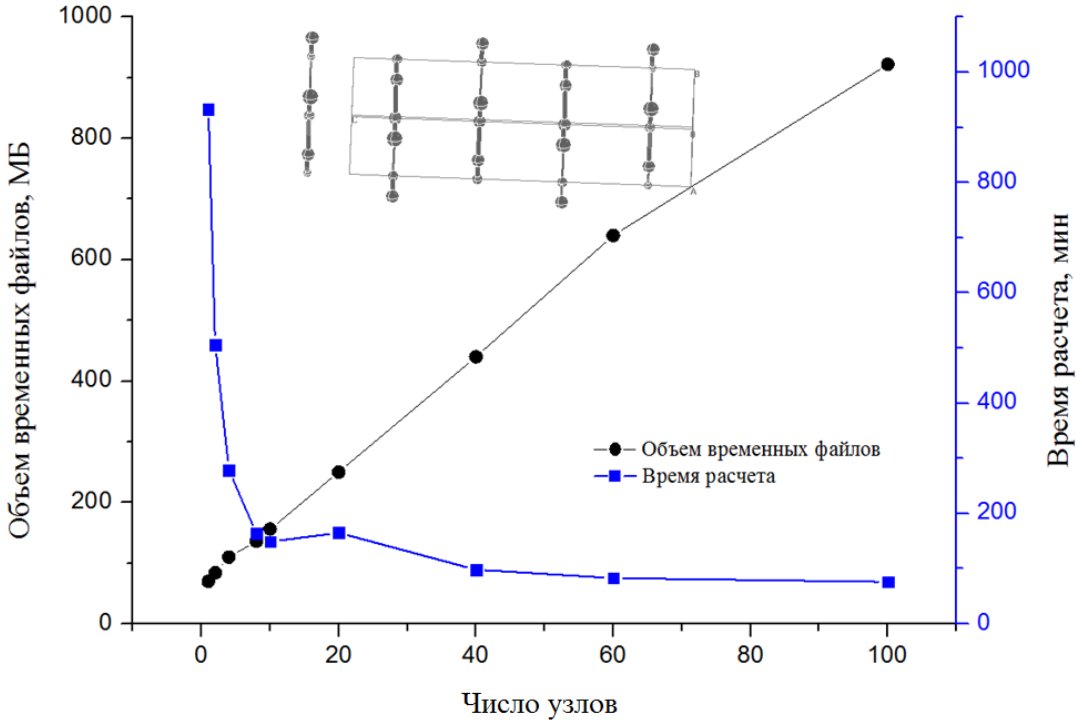


Рис. 1. Результаты расчетов кристалла силицированного графита (3D, 8 независимых атомов в ячейке, высокая симметрия)

Анализ объема временных файлов показывает, что эта величина для решения спектральной задачи оказывается существенно большей, чем для задачи оптимизации геометрии, хотя размер файлов растет в зависимости от числа узлов (табл. 4). Это связано с особенностями алгоритма расчета, который включает два последовательно идущих вычисления для каждого смещения атомов: вычисление фазы Бэрри и работа с временными файлами. Если первый процесс сопоставим с вычислительными затратами при оптимизации геометрии и способен масштабироваться, то второй характеризуется высокой интенсивностью обмена данными между оперативной и дисковой памятью. Поэтому увеличение выделенного числа узлов для расчета частот колебаний высокосимметричных структур с большим количеством атомов в ячейке ($>$ 15–20) скорее всего будет замедлять общее время расчета из-за пропорционально возрастающего обмена данными с файловым хранилищем и высокой загрузки внутренней сети суперкомпьютера. То есть, увеличение числа выделенных узлов для решения спектральной задачи и расчета колебательных спектров кристаллов имеет смысл только для сравнительно небольших объектов с числом атомов в ячейке не более 10–15. Небольшой выигрыш во времени расчета больших структур сопряжен с риском сбоя выполнения этой и других задач на суперкомпьютере.

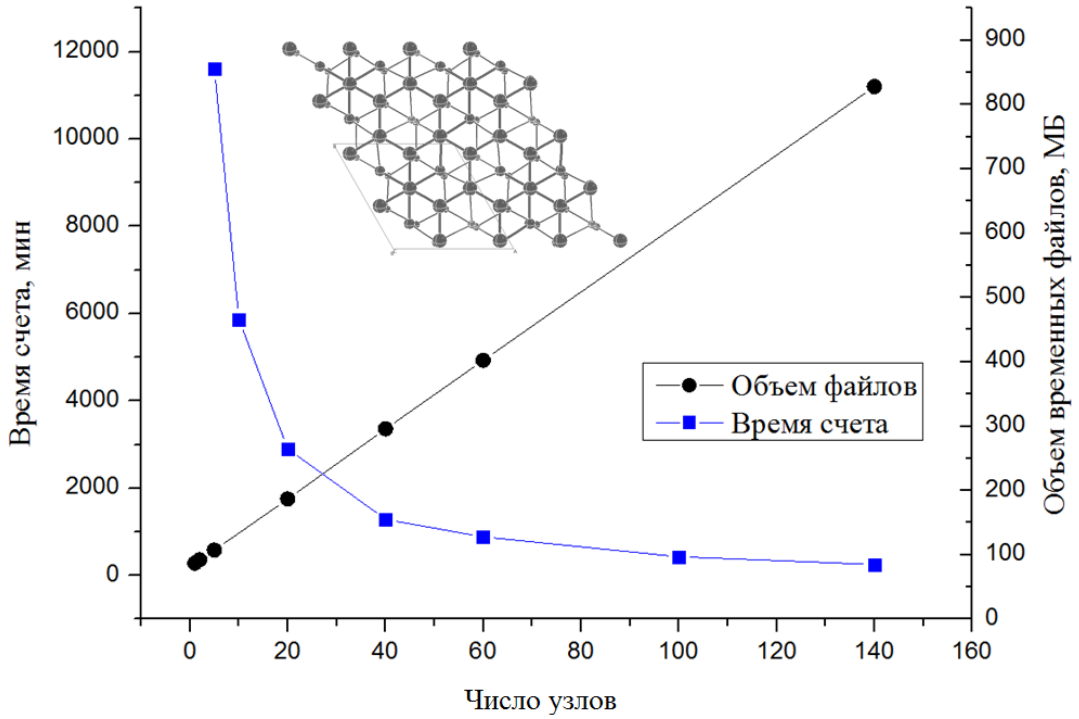


Рис. 2. Результаты расчетов слоя силицированного графита (2D, 24 независимых атома в ячейке, отсутствие симметрии)

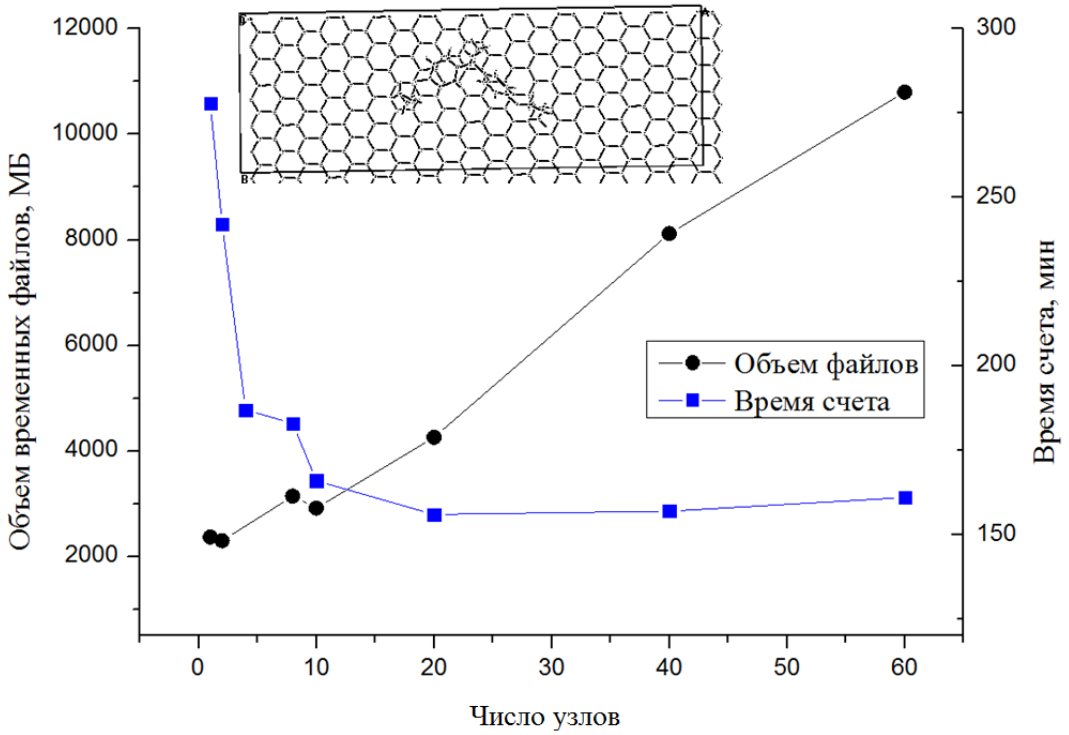


Рис. 3. Результаты расчетов комплекса бисфенола А, сорбированного на поверхности графена (2D, 273 атома в ячейке, низкая симметрия)

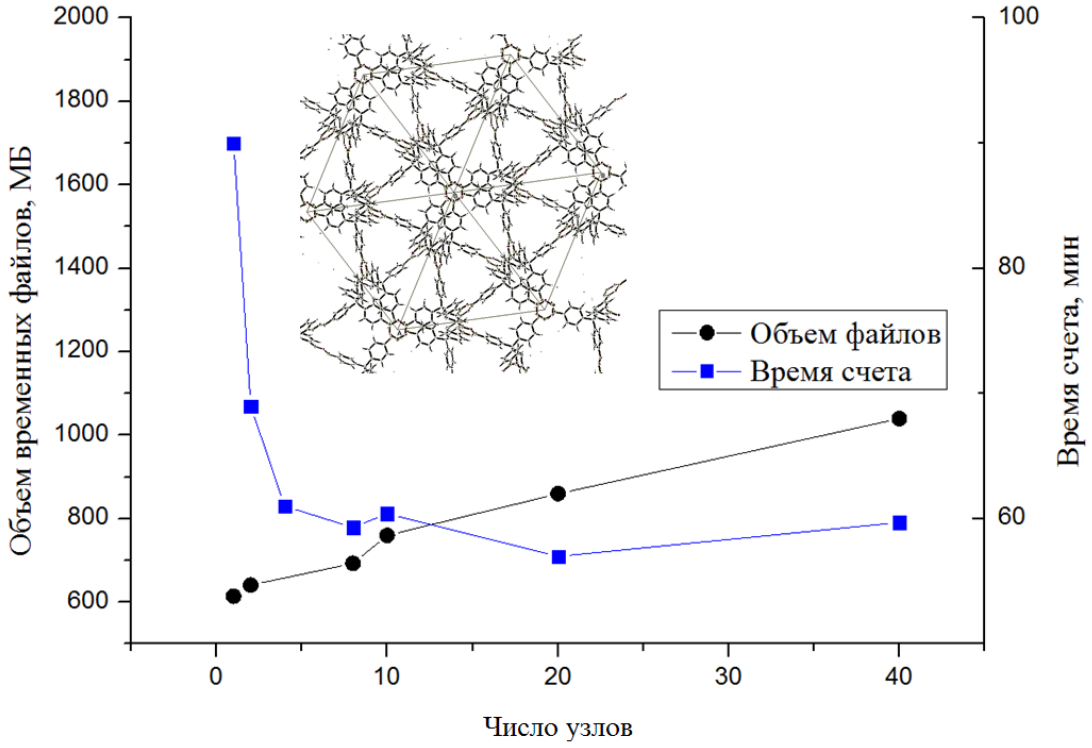


Рис. 4. Результаты расчетов кристалла ковалентного каркаса COF-102 (3D, 13 независимых атомов в ячейке, высокая симметрия)

Заключение

Таким образом, по результатам анализа временных затрат для решения различных квантовохимических задач, а именно: при поиске наиболее энергетически выгодной структуры кристалла или слоя и при расчете колебательных характеристик периодических структур на примере материалов с различным количеством атомов в элементарной ячейке, были выявлены следующие особенности и перспективы масштабируемости задач. Наибольшей масштабируемостью характеризуется оптимизация координат атомов в ячейке силицированного графита 2D, полученного из кристалла с высокой симметрией, в то время, как расчеты комплекса диглицидилового эфира бисфенола А на поверхности 2D графена за счет большого числа атомов в ячейке характеризуются большим объемом временных файлов и обладают существенно меньшими возможностями масштабирования. Структура кристаллического каркаса COF-102, с одной стороны, характеризуется высокой симметрией, а с другой — большим числом атомов в элементарной ячейке, и при увеличении числа узлов практически не демонстрирует выигрыша во времени расчета в процедуре оптимизации геометрии. Количество независимых атомов в элементарной ячейке исследуемой структуры материала влияет на размер временных файлов в ходе расчета, а время расчетов, в свою очередь, увеличивается с ростом числа используемых узлов. В связи с этим, для систем с большим количеством атомов, приходящихся на одну элементарную ячейку, или при использовании расширенных наборов атомных орбиталей – числа функций в базисном наборе, существенным оказывается увеличение объема временных файлов с увеличением числа используемых узлов, которое приводит к неоптимальному режиму расчетов и возможному сбою. Выяснено, что масштабируемость расчетов колебательных характеристик

молекулярных кристаллов, в частности, крупных кристаллических каркасов, существенно ниже, чем для расчетов, направленных на поиск наиболее энергетически выгодной структуры, независимо от числа атомов в элементарной ячейке вычисляемой системы. Вследствие этого, для систем с большим числом атомов в ячейке и долгим временем расчета оказывается нерациональным увеличение числа используемых узлов больше четырех в связи с удлинением временного интервала активной работы с файловым хранилищем при записи временных файлов и связанным с этим увеличением вероятности сбоя параллельно идущих расчетов.

Работа выполнена при финансовой поддержке поддержке гранта РНФ № 22-13-00170.

Литература

1. Xiang F., Zhang Z., Zuo Y., Tao F. Digital Twin Driven Green Material Optimal-Selection towards Sustainable Manufacturing // Procedia CIRP. 2019. Vol. 81, no. 1. P. 1290–1294. DOI: 10.1016/j.procir.2019.04.015.
2. Noel Y., D'Arco P., Demichelis R., *et al.* On the use of symmetry in the ab initio quantum mechanical simulation of nanotubes and related materials // Journal of Computational Chemistry. 2010. Vol. 31, no. 4. P. 855–862. DOI: 10.1002/jcc.21370.
3. Civalleri B., D'Arco P., Orlando R., *et al.* Hartree-Fock geometry optimisation of periodic systems with the CRYSTAL code // Chemical Physics Letters. 2001. Vol. 348, no. 1. P. 131–138. DOI: 10.1016/s0009-2614(01)01081-8.
4. Dovesi R., Pascale F., Civalleri B., *et al.* The CRYSTAL code, 1976–2020 and beyond, a long story // Journal of Chemical Physics. 2020. Vol. 152, no. 1. P. 204111. DOI: 10.1063/5.0004892.
5. Бандура А., Эварестов Р. Неэмпирические расчеты кристаллов в атомном базисе с использованием интернет-сайтов и параллельных вычислений. Санкт-Петербург: Издательство Санкт-Петербургского университета, 2004. 228 с.
6. Pascale F., Zicovich-Wilson C., Gejo F.L., *et al.* The calculation of vibrational frequencies of crystalline compounds and its implementation in the CRYSTAL code // Chemical Physics Letters. 2004. Vol. 25, no. 6. P. 888–897. DOI: 10.1002/jcc.20019.
7. Lohse M.S., Bein T. Covalent Organic Frameworks: Structures, Synthesis, and Applications // Advanced Functional Materials. 2018. Vol. 28, no. 33. P. 1705553. DOI: 10.1002/adfm.201705553.
8. Биленко Р., Долганина Н., Иванова Е., Рекачинский А. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 15–30. DOI: 10.14529/cmse220102.

Юшина Ирина Дмитриевна, к.х.н., научный сотрудник лаборатории многомасштабного моделирования многокомпонентных функциональных материалов, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Матвейчук Юрий Васильевич, к.х.н., научный сотрудник лаборатории многомасштабного моделирования многокомпонентных функциональных материалов, Южно-Уральский

государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Барташевич Екатерина Владимировна, д.х.н., главный научный сотрудник лаборатории многомасштабного моделирования многокомпонентных функциональных материалов, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse220304

SCALABILITY OF QUANTUM CHEMICAL CALCULATIONS OF CRYSTALLINE MATERIALS USING "TORNADO" SUPERCOMPUTER IN SOUTH URAL STATE UNIVERSITY

© 2022 I.D. Yushina, Yu.V. Matveychuk, E.V. Bartashevich

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: iushinaid@susu.ru, matveichukyv@susu.ru, bartashevichev@susu.ru

Received: 17.08.2022

Technical results of multiscale modeling of crystalline materials of different structure and composition has been presented and discussed. The scalability of different types of quantum chemical calculations using resources of «Tornado» supercomputer has been studied. Such tasks are extremely relevant in the field of modeling of structure and properties of crystals and carbon materials. The range of scalability has been reported for the systems of different size, composition, symmetry and level of modeling in the framework of density functional theory with atomic basis sets. A list of recommendations has been formulated presenting the optimal parameters for different types of material science tasks on the way to digital twin design. Analysis of calculation time for different systems and calculation types has been performed. A list of simple systems such as 3D silicon-substituted graphite as well as complex systems consisting of carbon surfaces (nanotube and graphene layer) and Bisphenol A diglycidyl ether as molecule on a surface. It is revealed that for large systems the most critical condition is the increase of the size of temporary files with the increase of number of nodes leading to possible failure of the calculation. It was shown that the scalability of the calculations of vibration properties of crystals is significantly lower than of the search of energetically preferable structure no matter how many atoms are located in the elementary part of the cell of computed structure.

Keywords: digital twin, quantum chemical calculations, functional materials, scalability.

FOR CITATION

Yushina I.D., Matveychuk Yu.V., Bartashevich E.V. Scalability of Quantum Chemical Calculations of Crystalline Materials Using "Tornado" Supercomputer in South Ural State University. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 3. P. 59–68. (in Russian) DOI: 10.14529/cmse220304.

References

1. Xiang F., Zhang Z., Zuo Y., Tao F. Digital Twin Driven Green Material Optimal-Selection towards Sustainable Manufacturing. Procedia CIRP. 2019. Vol. 81, no. 1. P. 1290–1294. DOI: 10.1016/j.procir.2019.04.015.

2. Noel Y., D'Arco P., Demichelis R., *et al.* On the use of symmetry in the ab initio quantum mechanical simulation of nanotubes and related materials. Journal of Computational Chemistry. 2010. Vol. 31, no. 4. P. 855–862. DOI: 10.1002/jcc.21370.

3. Civalleri B., D'Arco P., Orlando R., *et al.* Hartree-Fock geometry optimisation of periodic systems with the CRYSTAL code. Chemical Physics Letters. 2001. Vol. 348, no. 1. P. 131–138. DOI: 10.1016/s0009-2614(01)01081-8.

4. Dovesi R., Pascale F., Civalleri B., *et al.* The CRYSTAL code, 1976–2020 and beyond, a long story. Journal of Chemical Physics. 2020. Vol. 152, no. 1. P. 204111. DOI: 10.1063/5.0004892.

5. Bandura A., Evarestov R. Nonempirical methods of modeling of crystals with atomic basis set using parallel calculations. Saint Petersburg: Publishing of Saint Petersburg University, 2004. 228 p. (in Russian).

6. Pascale F., Zicovich-Wilson C., Gejo F.L., *et al.* The calculation of vibrational frequencies of crystalline compounds and its implementation in the CRYSTAL code. Chemical Physics Letters. 2004. Vol. 25, no. 6. P. 888–897. DOI: 10.1002/jcc.20019.

7. Lohse M.S., Bein T. Covalent Organic Frameworks: Structures, Synthesis, and Applications. Advanced Functional Materials. 2018. Vol. 28, no. 33. P. 1705553. DOI: 10.1002/adfm.201705553.

8. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC Resources of South Ural State University. Parallel Computational Technologies. PCT 2022. Communications in Computer and Information Science. Springer, 2022. DOI: 10.1007/978-3-031-11623-0_4.

УДК 004.272.25, 004.421, 004.032.24 DOI: 10.14529/cmse220305

ПАРАЛЛЕЛЬНЫЙ АЛГОРИТМ ВОССТАНОВЛЕНИЯ СЕНСОРНЫХ ДАННЫХ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ ДЛЯ МНОГОЯДЕРНОГО ПРОЦЕССОРА

© 2022 М.Л. Цымблер[1], А.Н. Полуянов[2], Я.А. Краева[1]

[1]*Южно-Уральский государственный университет*
(454080, Челябинск, пр. им. В.И. Ленина, д. 76),
[2]*Институт математики им. С.Л. Соболева СО РАН (644043 , Омск, ул. Певцова, д. 13)*
E-mail: mzym@susu.ru, andrey.poluyanov@gmail.com, kraevaya@susu.ru

Поступила в редакцию: 30.07.2022

В настоящее время во многих предметных областях обработка сенсорных данных в режиме реального времени связана с необходимостью синтеза значения соответствующего временного ряда, которое было пропущено ввиду технического сбоя или человеческого фактора. В данной статье предлагается параллельный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени для многоядерного процессора. Алгоритм использует набор опорных временных рядов, которые имеют семантическую связь с исходным рядом. Алгоритм применяет следующую эвристику: если в опорных рядах имеют место повторяющиеся (схожие) подпоследовательности, то в ряде, содержащем пропущенное значение, повторяющиеся подпоследовательности возникают в тех же временны́х интервалах. Образцами поиска для каждого опорного ряда полагаются подпоследовательности заданной длины, оканчивающиеся в момент пропуска значения в исходном ряде. Схожесть подпоследовательностей с образцом определяется на основе меры DTW (Dynamic Time Warping), имеющей квадратичную вычислительную сложность относительно длины подпоследовательности. Применяется техника нижних границ схожести, позволяющая отбрасывать подпоследовательности, заведомо непохожие на образец, без вычисления DTW. Нижние границы имеют меньшую, чем у DTW сложность, и вычисляются параллельно. Восстановленное значение вычисляется как среднее арифметическое последних элементов найденных интервалов. В вычислительных экспериментах предложенный алгоритм демонстрирует высокую точность восстановления в сравнении с аналогами и быстродействие, приемлемое для применения алгоритма в режиме реального времени.

Ключевые слова: временной ряд, восстановление пропущенных значений, параллельный алгоритм, многоядерный процессор, DTW, отбрасывание по нижним границам.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Цымблер М.Л., Полуянов А.Н., Краева Я.А. Параллельный алгоритм восстановления сенсорных данных в режиме реального времени для многоядерного процессора // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 3. С. 69–90. DOI: 10.14529/cmse220305.

Введение

В настоящее время обработка сенсорных данных в режиме реального времени возникает в широком спектре приложений, например, интеллектуальное управление зданиями [1], цифровые двойники [2], мониторинг показателей функциональной диагностики организма человека [3], моделирование климата [4] и др. В указанных приложениях, как правило, недопустимы пропущенные значения, которые возникат ввиду отказов сенсоров или человеческого фактора, и они должны быть незамедлительно заменены на правдоподобные синтезированные значения. В соответствии с этим является актуальной задача разработки алгоритмов восстановления пропущенных значений в потоковых сенсорных данных, которые обеспечивают высокие точность и быстродействие восстановления.

В данной работе представлен параллельный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени для многоядерного процессора. Алгоритм использует поиск подпоследовательностей ряда, похожих на заданную подпоследовательность-образец в смысле меры схожести DTW (Dynamic Time Warping, динамическая трансформация времени) [5]. На сегодня DTW признается научным сообществом одной из лучших мер схожести временных рядов для многих предметных областей [6]. Однако мера DTW имеет квадратичную вычислительную сложность относительно длины ряда и определяется рекуррентными формулами, что требует нетривиального подхода к распараллеливанию ее вычисления. Работа продолжает исследование авторов, начатое в статье [7]. По сравнению с предшествующей работой существенно увеличена производительность алгоритма за счет внедрения техники, позволившей отбрасывать без вычисления DTW большее количество подпоследовательностей, заведомо непохожих на образец поиска, а также проведены более масштабные вычислительные эксперименты.

Статья организована следующим образом. Раздел 1 содержит обзор работ, наиболее близких к выполненному исследованию. В разделе 2 приводится формальная постановка задачи. В разделе 3 представлен предлагаемый параллельный алгоритм. В разделе 4 описаны вычислительные эксперименты по исследованию эффективности предложенного алгоритма. Заключение резюмирует полученные результаты исследования и обозначает направление будущих работ.

1. Обзор работ

Создание эффективных методов и алгоритмов восстановления пропущенных данных является одной из наиболее актуальных проблем обработки временных рядов [8]. В данном разделе кратко рассмотрены только те работы, которые наиболее близки проведенному исследованию и затрагивают следующие аспекты: применение для восстановления ряда принципа ближайших соседей, меры схожести DTW, опорных рядов (коррелирующих с исходным рядом), а также использование параллельных вычислений.

В работе [9] Батиста (Batista) и др. предложили метод восстановления значений временного ряда kNNI (k-Nearest Neighbor Imputation) на основе ближайших соседей. Пусть имеется объект с множеством атрибутов, один из которых (обозначим его как A) имеет пропущенные значения. Метод предписывает сначала найти k «соседей» — объектов, имеющих схожие значения в атрибутах, отличных от A (без конкретизации меры схожести). Затем отсутствующее значение синтезируется на основе значений в атрибуте A у соседей. В работе [10] Троянская (Troyanskaya) и др. предложили использование взвешенных ближайших соседей, когда вес соседа пропорционален схожести с образцом поиска в смысле евклидовой метрики.

В работе [11] Хсю (Hsu) и др. предложили восстановление отсутствующих значений, основанный на использовании принципа k ближайших соседей и меры DTW. Алгоритм используется для восстановления отсутствующих значений во временном ряде, который представляет собой значения уровня экспрессии генов, полученные с помощью ДНК-микрочипов в серии экспериментов.

В работе [12] Фан (Phan) и др. применили DTW для восстановления пропущенных значений в многомерном временном ряде. Восстановление выполняется отдельно для каждого ряда-координаты следующим образом. Подпоследовательность ряда, которая начинается непосредственно после промежутка пропущенных значений и имеет ту же длину, что и

указанный промежуток, объявляется образцом поиска. Далее в части ряда, следующей после образца, выполняется поиск подпоследовательности, которая по длине равна образцу и является самой похожей на образец в смысле меры DTW. Подпоследовательность, располагающаяся непосредственно перед найденной и имеющая ту же длину, используется для поэлементного заполнения пропущенного промежутка.

Те же авторы в работе [13] представили алгоритм DTWBI (DTW-Based Imputation) для восстановления пропущенных значений в одномерном временном ряде. Восстанавливаемый временной ряд подвергается преобразованию DDTW (Derivative DTW) [14]. Далее подпоследовательность ряда, находящаяся непосредственно перед промежутком из пропущенных значений и имеющая ту же длину, что и указанный промежуток, объявляется образцом поиска. Далее в части ряда, предшествующей образцу, выполняется поиск подпоследовательности, самой похожей на образец в смысле меры DTW. Подпоследовательность, расположенная после найденной и имеющая ту же длину, используется для поэлементного заполнения пропущенного промежутка.

В работе [15] Велленцон (Wellenzohn) и др. предложили алгоритм TKCM (Top-k Case Matching), использующий для восстановления пропущенных значений ряда набор опорных временных рядов (reference time series), которые имеют семантическую связь с исходным рядом. Примерами исходного и опорных временных рядов могут служить показания температурных датчиков, установленных в географически близких локациях [15]. Алгоритм применяет эвристику, согласно которой похожие ситуации в опорных рядах возникают в те же временные промежутки, что и в ряде, подлежащем восстановлению. В каждом из опорных рядов TKCM объявляет образцом поиска подпоследовательность, завершающуюся в момент возникновения пропуска в исходном ряде. Далее в каждом из опорных рядов алгоритм выполняет поиск k подпоследовательностей, которые являются ближайшими соседями выбранных образцов и не могут перекрывать друг друга. При поиске используется схема динамического программирования на основе евклидовой метрики, которая минимизирует целевую функцию суммарного отличия подпоследовательностей в опорных рядах от соответствующих образцов. Восстанавливаемое значение получается как среднее арифметическое точек исходного ряда, которые соответствуют завершающим точкам временных интервалов найденных ближайших соседей. Алгоритм обеспечивает хорошую точность восстановления, когда отсутствуют блоки значений. Однако это достигается за счет прямо пропорциональной зависимости быстродействия алгоритма от его основных параметров: длина образца, количество опорных временных рядов, число ближайших соседей.

В обзоре и экспериментальном сравнении двенадцати современных алгоритмов восстановления пропущенных значений временных рядов [8] Хаяти (Khayati) и др. отмечают, что в настоящее время, по-видимому, отсутствуют успешные попытки использовать аппаратное обеспечение для ускорения алгоритмов восстановления отсутствующих значений в больших временных рядах.

В данном исследовании предлагается параллельный алгоритм восстановления пропущенных значений временного ряда для многоядерного процессора. Подобно алгоритму TKCM, наш алгоритм использует поиск ближайших соседей в опорных временных рядах. Однако при поиске вместо схемы динамического программирования и евклидовой метрики используется мера схожести DTW, которая имеет квадратичную вычислительную сложность, но в общем случае лучше отражает схожесть формы образца поиска и подпоследовательности ряда [6]. При этом применяется каскад нижних границ схожести [16]: под-

последовательности, заведомо непохожие на запрос, отбрасываются без вычисления DTW, что существенно сокращает объем вычислений [6]. В реализации используется идея предварительного параллельного вычисления нижних границ, предложенная нами ранее в работе [17]. Указанные разработки обуславливают возможность применения нашего алгоритма в режиме реального времени.

2. Формальные определения и обозначения

Потоковый временной ряд (streaming time series) представляет собой набор вещественных значений, поступающих последовательно одно за другим в режиме реального времени:

$$T = (\ldots, t_{n-2}, t_{n-1}, t_n), \quad t_i \in \mathbb{R}. \tag{1}$$

Мы полагаем, что n-элементное окно ряда T может быть размещено в оперативной памяти и n имеет порядок десятков-сотен тысяч элементов.

Значение элемента ряда t_n в текущий момент времени пропущено, что обозначается как t_n=`NULL`. Решаемая нами задача состоит в том, чтобы вместо пропущенного значения в режиме реального времени вычислить синтетическое значение $\tilde{t}_n$, которое как можно меньше отличалось бы от t_n.

Предлагаемый в данной работе алгоритм синтеза значения $\tilde{t}_n$ основан на поиске в ряде T подпоследовательности C, которая является самой похожей на заданный образец поиска Q в смысле меры схожести DTW (Dynamic Time Warping) [5]. Ниже приводятся соответствующие формальные определения с использованием нотации из работы [16].

Подпоследовательностью (subsequence) временного ряда T, имеющей длину m, называют непрерывное подмножество T из m элементов, начиная с позиции i:

$$T_{i,m} = (t_i, \ldots, t_{i+m-1}), \quad 1 \leqslant m \ll n, \quad 1 \leqslant i \leqslant n - m + 1. \tag{2}$$

Множество всех подпоследовательностей ряда T, имеющих длину m, обозначается как S_T^m.

Пусть имеются две подпоследовательности ряда T: $C = (c_1, \ldots, c_m)$ и $Q = (q_1, \ldots, q_m)$. Тогда DTW-расстояние между C и Q определяется следующим образом [5]:

$$\begin{gathered} \mathrm{DTW}(C, Q) = D(m, m), \\ D(i, j) = (c_i - q_j)^2 + \min \begin{cases} D(i-1, j) \\ D(i, j-1) \\ D(i-1, j-1) \end{cases}, \\ D(0, 0) = 0, \quad D(i, 0) = D(0, j) = +\infty, \quad 1 \leqslant i \leqslant m, \quad 1 \leqslant j \leqslant m. \end{gathered} \tag{3}$$

В формуле (3) $D \in \mathbb{R}^{m \times m}$ является *матрицей трансформации*, которая задает выравнивание между подпоследовательностями C и Q друг относительно друга. В данной матрице строится *путь трансформации*, представляющий собой набор смежных элементов матрицы, который начинается и заканчивается в диагонально противоположных крайних элементах и устанавливает соответствие между C и Q, минимизируя общее расстояние между ними.

Подпоследовательность $C \in S_T^m$ называется *самой похожей (best match subsequence)* на заданный *запрос (query, образец поиска)* Q, если

$$\mathrm{DTW}(Q, C) \leqslant \mathrm{DTW}(Q, T_{i,m}), \quad 1 \leqslant i \leqslant n - m + 1. \tag{4}$$

Прямолинейный поиск самой похожей подпоследовательности (вычисление DTW-расстояния для всех подпоследовательностей ряда и нахождение среди них глобального минимума) не подходит для режима реального времени, поскольку DTW имеет вычислительную сложность $O(m^2)$ [6]. В силу этого нами используется *техника нижних границ (lower bounding)* [6], которая позволяет отбрасывать подпоследовательности, заведомо непохожие на запрос, без вычисления DTW, и существенно сократить объем вычислений.

Нижняя граница (lower bound) представляет собой симметричную неотрицательную функцию $\mathrm{LB} : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ с вычислительной сложность меньше $O(m^2)$, которая является строго доказанным порогом DTW-расстояния между подпоследовательностью ряда и запросом:

$$\forall\, T_{i,\,m} \in S_T^m,\, Q \quad \mathrm{LB}(Q, T_{i,\,m}) \leqslant \mathrm{DTW}(Q, T_{i,\,m}). \tag{5}$$

Техника нижних границ заключается в следующем [6]. Обозначим локальный минимум DTW-расстояния от подпоследовательностей ряда до запроса как *bsf (best-so-far)* и инициализируем указанную переменную значением $+\infty$. Затем будем выполнять движение скользящего окна длины m в ряде T слева направо с шагом 1, вычисляя при этом нижнюю границу текущей подпоследовательности. Если результат превышает bsf, то в силу (5) значение DTW-расстояния между текущей подпоследовательностью и запросом также превысит bsf. Следовательно, данная подпоследовательность заведомо непохожа на запрос и может быть отброшена без вычисления DTW. Если вычисленное DTW-расстояние меньше локального минимума, то оно обновляет значение bsf. Формальная запись описанной выше техники выглядит следующим образом:

$$\begin{cases} bsf_{(0)} & = +\infty \\ bsf_{(i)} & = \min\Big(bsf_{(i-1)}, \begin{cases} +\infty & , \mathrm{LB}(Q, T_{i,\,m}) > bsf_{(i-1)} \\ \mathrm{DTW}(Q, T_{i,\,m}) & , otherwise \end{cases}\Big), \end{cases} \tag{6}$$

где нижний индекс в скобках означает номер шага i $(0 \leqslant i < n - m + 1)$.

Для применения техники нижних границ требуется, чтобы подпоследовательность и запрос были подвергнуты z-нормализации [16]. Z-нормализацией ряда $T = (t_1, \ldots, t_m)$ является ряд $\hat{T} = (\hat{t}_1, \ldots, \hat{t}_m)$, элементы которого вычисляются следующим образом:

$$\hat{t}_i = \frac{t_i - \mu}{\sigma}, \quad \mu = \frac{1}{m}\sum_{i=1}^{m} t_i, \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m} t_i^2 - \mu^2. \tag{7}$$

Для повышения эффективности отбрасывания заведомо непохожих подпоследовательностей нижние границы применяют каскадом [16] (одна за другой в порядке возрастания их вычислительной сложности). В нашем исследовании применяются три наиболее известные нижние границы $\mathrm{LB_{Kim}FL}$ [18], $\mathrm{LB_{Keogh}EQ}$ [19] и $\mathrm{LB_{Keogh}EC}$ [16].

Нижняя граница схожести $\mathrm{LB_{Kim}FL}$ представляет собой квадрат Евклидова расстояния между первой и последней парами точек запроса и подпоследовательности:

$$\mathrm{LB_{Kim}FL}(Q, C) = (\hat{q}_1 - \hat{c}_1)^2 + (\hat{q}_m - \hat{c}_m)^2. \tag{8}$$

Нижняя граница $\mathrm{LB_{Keogh}EC}$ показывает схожесть между *оболочкой (envelope) E* запроса и подпоследовательностью и вычисляется следующим образом:

$$\mathrm{LB_{Keogh}EC}(Q,C) = \sum_{i=1}^{m} \begin{cases} (\hat{c}_i - u_i)^2 & , \ \hat{c}_i > u_i \\ (\hat{c}_i - \ell_i)^2 & , \ \hat{c}_i < \ell_i \\ 0 & , \ otherwise \end{cases} . \tag{9}$$

В формуле (9) последовательности $U = (u_1, \ldots, u_m)$ и $L = (\ell_1, \ldots, \ell_m)$ обозначают *верхнюю* и *нижнюю границы оболочки* запроса Q соответственно, которые вычисляются следующим образом:

$$\begin{aligned} u_i &= \max(\hat{q}_{i-r}, \ldots, \hat{q}_{i+r}), \\ \ell_i &= \min(\hat{q}_{i-r}, \ldots, \hat{q}_{i+r}), \end{aligned} \tag{10}$$

где параметр r $(1 \leqslant r \leqslant m)$ представляет собой *ограничение полосы Сако—Чиба (Sakoe–Chiba band)* [6], которое показывает, что путь трансформации не может отклоняться более чем на r ячеек от диагонали матрицы трансформации.

Нижняя граница $\mathrm{LB_{Keogh}EQ}$ представляет собой Евклидово расстояние между запросом Q и оболочкой подпоследовательности C, то есть по сравнению с $\mathrm{LB_{Keogh}EC}$ роли запроса и подпоследовательности меняются местами:

$$\mathrm{LB_{Keogh}EQ}(Q,C) = \mathrm{LB_{Keogh}EC}(C,Q). \tag{11}$$

Каскад может быть дополнен другими нижними границами, например, $\mathrm{LB_{Yi}}$, $\mathrm{LB_{PAA}}$ [6]. Техника нижних границ позволяет отбросить до 99% вычислений DTW [16].

Завершая данный раздел, отметим, что описанные выше построения могут быть тривиально обобщены на случай поиска k подпоследовательностей ряда, наиболее похожих на образец [16], где k — наперед заданный параметр.

3. Параллельный алгоритм восстановления пропусков

В данном разделе представлен новый параллельный алгоритм для многоядерного процессора, который выполняет восстановление пропущенного значения потокового временного ряда в режиме реального времени. Ниже в разделе 3.1 описана общая схема восстановления, используемая алгоритмом. В разделе 3.2 представлены структуры данных алгоритма. Раздел 3.3 содержит описание принципов реализации.

3.1. Общая схема восстановления

Алгоритм применяет опорные временные ряды и следующую эвристику, примененные ранее в алгоритме TKCM [15]: если в опорных рядах имеют место повторяющиеся (похожие) подпоследовательности, то в ряде, содержащем пропущенное значение, повторяющиеся подпоследовательности возникают в тех же временны́х интервалах.

Пусть $R^1, \ldots, R^d$ $(d > 1)$ — опорные потоковые временные ряды для ряда T. Это означает, что в каждом опорном ряде отсутствуют `NULL`-значения, и его элементы получены в те же моменты времени, что и элементы ряда T с соответствующими порядковыми номерами. Пусть в данном списке ряды упорядочены по убыванию корреляции с T, например, по убыванию абсолютного значения коэффициента корреляции Пирсона. Как и в случае исходного ряда, мы предполагаем, что совокупность n-элементных окон опорных рядов размещена в

оперативной памяти. Алгоритм выполняет следующие шаги: поиск по образцу, скоринг и реконструкция.

На шаге *поиска по образцу* сначала для каждого опорного ряда R^i определяется образец, состоящий из m последних по времени элементов ряда (где $1 \leqslant m \ll n$ — параметр алгоритма): $Q^i = R^i_{n-m+1,\,m}$. Далее для каждого опорного ряда R^i в его окне с первого по $(n-2m)$-й элемент выполняется поиск подпоследовательности, которая имеет длину m и является самой похожей на образец Q^i в смысле меры DTW. Попутно при выполнении поиска вычисляется DTW-расстояние от запроса до каждой подпоследовательности опорного ряда, за исключением подпоследовательностей, заведомо не похожих на запрос, для которых сохраняется значение $+\infty$.

На шаге *скоринга*, используя вычисленные на предыдущем шаге DTW-расстояния от запросов до подпоследовательностей опорных рядов, определяется множество $kNNset$ подпоследовательностей исходного ряда, применяемых для реконструкции. Данное множество состоит из k не пересекающихся друг с другом подпоследовательностей длины m (где $1 \leqslant k < \lceil n/m \rceil$ — параметр алгоритма), которые являются наиболее значимыми для восстановления пропущенного значения. Значимость подпоследовательности определяется нами как функция от DTW-расстояний соответствующих подпоследовательностей в опорных рядах, подсчитанных на шаге поиска по образцу.

На шаге *реконструкции* восстановленное значение вычисляется как среднее арифметическое последних элементов подпоследовательностей, найденных на шаге скоринга.

3.2. Структуры данных

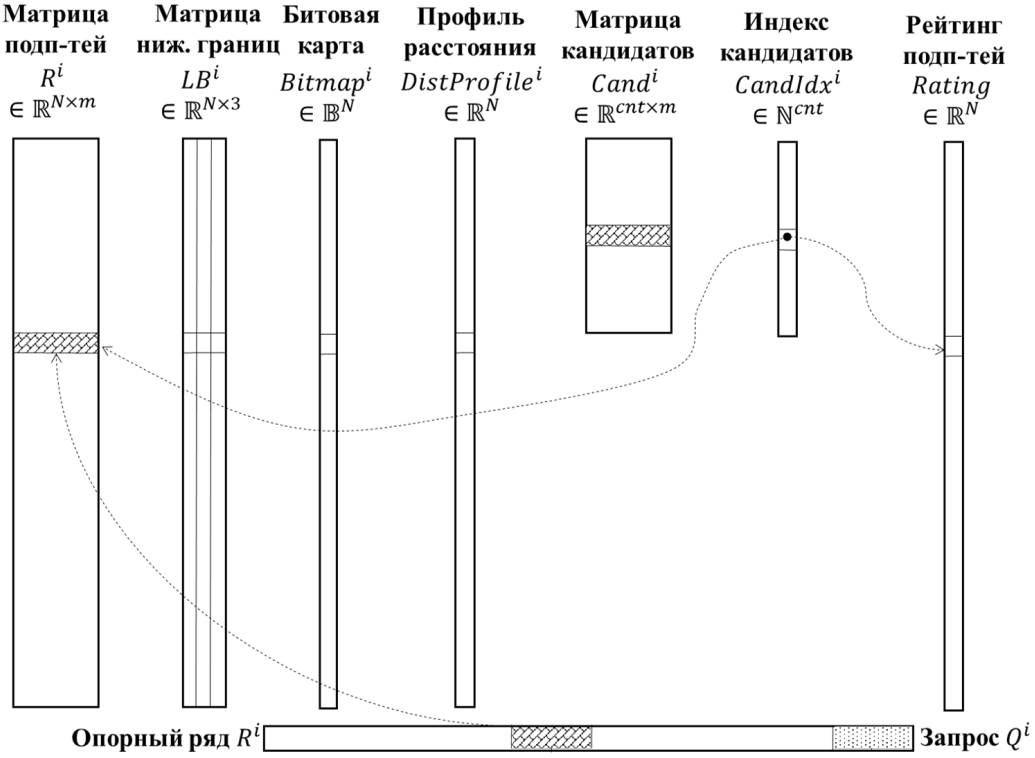


Рис. 1. Структуры данных алгоритма

Структуры данных нашего алгоритма представлены на рис. 1. Каждый опорный временной ряд хранится в виде матрицы подпоследовательностей $R^i \in \mathbb{R}^{N\times m}$, где число

$N = n - m$ представляет собой количество подпоследовательностей, имеющих длину m, исключая запрос. Для каждого опорного ряда определяются следующие структуры данных: матрица нижних границ, битовая карта, профиль расстояния, матрица и индекс кандидатов. Помимо указанных структур данных, для выполнения реконструкции хранится рейтинг интервалов исходного временного ряда.

Матрица нижних границ $LB^i \in \mathbb{R}^{N \times lb_{num}}$ хранит значения нижних границ, вычисляемых по формулам (8)–(11), где lb_{num} означает количество нижних границ в каскаде (в нашем исследовании $lb_{num} = 3$). Отметим, что в оригинальном алгоритме [16] нижние границы вычисляются при движении скользящего окна, и при этом каждая следующая нижняя граница каскада вычисляется только в том случае, если текущая подпоследовательность не была отброшена как заведомо непохожая на запрос. Очевидно, что такие вычисления не могут быть распараллелены в силу наличия между ними зависимостей по данным. В предлагаемом нами подходе все нижние границы для всех подпоследовательностей ряда вычисляются одномоментно до начала движения скользящего окна. Несмотря на избыточность таких вычислений, они выполняются однократно и при этом могут быть распараллелены ввиду отсутствия зависимостей по данным между нижними границами, что в итоге дает выигрыш в производительности. Платой за получаемое быстродействие, очевидно, является необходимость хранения описанной матрицы в памяти.

Битовая карта представляет собой вектор $Bitmap^i \in \mathbb{B}^N$, в котором для каждой подпоследовательности опорного временного ряда хранится конъюнкция результатов сравнения bsf и каждой из нижних границ (8)–(11) для запроса Q^i:

$$Bitmap^i(j) = \bigwedge_{s=1}^{lb_{num}} \big(LB^i(j,s) < bsf\big). \tag{12}$$

Таким образом, если какая-либо нижняя граница каскада превысит значение bsf, то битовая карта содержит `FALSE` и показывает, что соответствующая подпоследовательность заведомо не похожа на запрос.

Профиль расстояния представляет собой вектор $DistProfile^i \in \mathbb{R}^N$, в котором хранятся DTW-расстояния между подпоследовательностями опорного ряда R^i и запросом Q^i. Для подпоследовательностей, заведомо не похожих на запрос, в профиле расстояния хранится значение $+\infty$.

Матрица кандидатов $Cand^i \in \mathbb{R}^{cnt^i \times m}$ хранит подпоследовательности, которые не были отброшены как заведомо непохожие на образец Q^i, где количество таких подпоследовательностей cnt^i определяется после применения каскада нижних границ. Данная матрица используется для параллельного вычисления DTW-расстояний между каждой из ее строк и запросом.

Индекс кандидатов $CandIndex^i \in \mathbb{N}^{cnt^i}$ хранит номера соответствующих строк $Cand^i$ в опорном ряде R^i.

Рейтинг подпоследовательностей $Rating \in \mathbb{R}^N$ представляет собой вектор, который хранит значимость подпоследовательностей длины m в потоковом n-элементном окне для последующего поиска k наиболее значимых из них и выполнения реконструкции отсутствующего значения t_n.

3.3. Реализация

В данном разделе описаны принципы параллельной реализации шагов поиска и скоринга (шаг реконструкции является алгоритмически тривиальным). Параллелизм для многоядерного процессора реализован на основе технологии программирования OpenMP [20]. В модели OpenMP приложение рассматривается как процесс — набор инструкций, исполняемых последовательно. Процесс может запустить (*fork*) заданное количество параллельно исполняемых нитей. Нити разделяют память процесса и в то же время для хранения собственных данных имеют приватную память. Среда исполнения приложения автоматически назначает нитям различные ядра процессора. Формирование параллельных нитей осуществляется вставкой директивы компилятора `#pragma` в исходный код. Одним из наиболее типичных случаев применения данной директивы является распараллеливание цикла **for** с помощью `#pragma omp parallel for`, когда каждая из нитей обрабатывает собственный диапазон значений счетчика цикла.

Алг. 1 SEARCH(IN: R, Q; OUT: $DistProfile$)

1: $LB \leftarrow$ CALCLBS(R, Q)
2: $idx \leftarrow \arg\min_{i \in 1..N} \max_{j \in 1..lb_{num}} LB(i,j)$; $bsf \leftarrow \mathrm{DTW}\big(\hat{Q}, R(idx, \cdot)\big)$
3: $Bitmap \leftarrow$ MAKEBITMAP(LB, bsf)
4: $DistProfile \leftarrow \overline{+\infty}$
5: **while** `TRUE` **do**
▷ Отбрасывание бесперспективных кандидатов с помощью нижних границ
6: #pragma omp parallel for
7: **for** $i \in 1..N$ **do**
8: $Bitmap(i) \leftarrow Bitmap(i)$ **and** $\bigwedge_{j=1}^{lb_{num}} \big(LB(i,j) < bsf\big)$
▷ Заполнение матрицы кандидатов и индекса кандидатов
9: $cnt \leftarrow 0$
10: **for** $i \in 1..N$ **do**
11: **if** $Bitmap(i) =$ `TRUE` **then**
12: $cnt \leftarrow cnt + 1$; $Cand(cnt) \leftarrow R(i, \cdot)$; $CandIndex(cnt) \leftarrow i$
13: **if** $cnt = 0$ **then**
14: **break**
▷ Вычисление DTW-расстояний для кандидатов и улучшение bsf
15: $s \leftarrow 1$
16: **while** `TRUE` **do**
17: $left \leftarrow p \cdot (s-1) + 1$; $right \leftarrow \min(cnt, p \cdot s)$
18: #pragma omp parallel for reduction(min: $dist$)
19: **for** $i \in left..right$ **do**
20: $dist \leftarrow \mathrm{DTW}(\hat{Q}, Cand(i, \cdot))$
21: $DistProfile(CandIndex(i)) \leftarrow dist$
22: $Bitmap(CandIndex(i)) \leftarrow$ `FALSE`
23: $bsf \leftarrow \min(bsf, dist)$; $s \leftarrow s + 1$
24: **if** $dist < bsf$ **or** $right = cnt$ **then**
25: **break**
26: **return** $DistProfile$

Поиск реализуется с помощью двух вложенных циклов: внешний цикл по опорным временным рядам и внутренний цикл по подпоследовательностям опорного временного ряда. Поскольку, как правило, количество подпоследовательностей в опорном ряде существенно больше количества опорных рядов, а последнее, в свою очередь, существенно меньше, чем количество нитей приложения, мы выполняем распараллеливание внутреннего цикла, чтобы обеспечить значимую нагрузку нитей. Далее для упрощения записи алгоритма мы опускаем внешний цикл по опорным временным рядам и не пишем в структурах данных соответствующий верхний индекс. За p обозначено количество нитей, на которых исполняется параллельное приложение. Псевдокод шага поиска представлен в Алг. 1. Алгоритм выполняется следующим образом.

Алг. 2 CalcLBs(IN: R, Q; OUT: LB)

```
1: #pragma omp parallel for
2: for i ∈ 1..N do
3:     R(i,·) ←zNormalize(R(i,·))
4:     LB(i,1) ← LB_KimFL(Q̂, R(i,·))
5:     LB(i,2) ← LB_KeoghEC(Q̂, R(i,·))
6:     LB(i,3) ← LB_KeoghEQ(Q̂, R(i,·))
7: return LB
```

Сначала параллельно нормализуются подпоследовательности и вычисляются нижние границы (см. строку 1 в Алг. 1 и Алг. 2). Затем выполняется инициализация порога bsf (см. строку 2). Начальное значение bsf выбирается нами следующим образом на основе эвристики, описанной в работе [16]. В указанной статье авторами проведены вычислительные эксперименты над 50 временными рядами из архива UCR [21], в которых для каждой нижней границы LB, упомянутой в разделе 2, вычислялось значение функции

$$\mathrm{Tightness}_{\mathrm{LB}}(A,B) = \frac{\mathrm{LB}(A,B)}{\mathrm{DTW}(A,B)} \tag{13}$$

для 100 тыс. случайных подпоследовательностей A и B, имеющих длину 256. Эксперименты показали [16], что, как правило, выполняется следующее неравенство:

$$\begin{gathered}\mathrm{Tightness}_{\mathrm{LB_{Kim}FL}}(A,B) < \mathrm{Tightness}_{\mathrm{LB_{Keogh}EQ}}(A,B) \leqslant \\ \leqslant \max\big(\mathrm{Tightness}_{\mathrm{LB_{Keogh}EQ}}(A,B), \mathrm{Tightness}_{\mathrm{LB_{Keogh}EC}}(A,B)\big) \leqslant 1.\end{gathered} \tag{14}$$

Исходя из соотношения (14) и определения нижней границы (5), в качестве начального значения для порога bsf мы выбираем DTW-расстояние от запроса до такой подпоследовательности ряда, на которой достигается минимум среди максимумов значений нижних границ:

$$bsf_{(0)} = \mathrm{DTW}(Q,C), \quad C = \arg \min_{T_{i,\,m} \in S_T^m} \max_{j \in 1..lb_{num}} \mathrm{LB}_j(Q, T_{i,m}). \tag{15}$$

В отличие от значения $bsf_{(0)} = +\infty$ в (6), это позволяет сократить количество рассматриваемых подпоследовательностей уже на ранней стадии.

После этого выполняется формирование битовой карты подпоследовательностей (см. строку 3 в Алг. 1 и Алг. 3). Сначала в соответствии с начальным значением порога bsf отбрасываются подпоследовательности, заведомо не похожие на запрос (см. строки 2–3

Алг. 3 МакеBitmap(IN: LB, bsf; OUT: $Bitmap$)

```
1:  match ← 0
2:  for i ∈ 1..N do
3:      Bitmap(i) ← ⋀_{j=1}^{lb_num} (LB(i, j) < bsf)
4:      if Bitmap(i) = TRUE then
5:          if match > 0 then
6:              right ← i
7:          else
8:              left ← i
9:          match ← match + 1
10:     if (Bitmap(i) = TRUE and match = m) or (Bitmap(i) = FALSE and match > 1) then
11:         idx ← arg min_{i∈left..right} max_{j∈1..lb_num} LB(i, j)
12:         for i ∈ {left..idx − 1} ∪ {idx + 1..right} do
13:             Bitmap(i) ← FALSE
14:         match ← 0
15: return Bitmap
```

в Алг. 3). Затем среди оставшихся кандидатов выполняется отбрасывание тривиальных совпадений (см. строки 4–14 в Алг. 3). Две подпоследовательности $T_{i,m}, T_{j,m} \in S_T^m$ являются *тривиальными совпадениями (trivial matches)* друг друга, если $|i - j| < m$ [22]. В каждой группе подпоследовательностей, представляющих тривиальные совпадения друг друга, мы оставляем одну подпоследовательность в соответствии с описанной выше эвристикой (см. формулу (15)), а остальные подпоследовательности группы отбрасываются.

Затем алгоритм поиска инициализирует профиль расстояния значениями $+\infty$ и итеративно уменьшает значение bsf, не вычисляя DTW-расстояния до подпоследовательностей, заведомо не похожих на запрос (строки 5–25). При этом сначала параллельно вычисляется битовая карта (строки 6–8), а затем формируются матрица кандидатов и индекс кандидатов (строки 10–12). Если подпоследовательности-кандидаты не найдены, то алгоритм останавливается (строки 13–14). Ситуация, когда после первого сканирования все подпоследовательности отброшены как заведомо непохожие на запрос, означает, что выбрано слишком жесткое ограничение полосы Сако—Чиба, и параметр r необходимо уменьшить.

Далее выполняется параллельное вычисление DTW-расстояния между каждой строкой матрицы кандидатов и запросом (строки 15–25). Вычисления производятся итеративно для групп по p строк, чтобы обеспечить баланс загрузки нитей. На каждой итерации мы автоматически получаем минимум DTW-расстояния и обновляем bsf и профиль расстояния, применяя параллельную свертку (конструкция `reduction` в директиве `#pragma`, строка 18). В элемент битовой карты, соответствующий обработанной подпоследовательности, записывается значение `FALSE`, чтобы обеспечить однократную обработку каждого кандидата. Описанная деятельность продолжается, пока не будет улучшено (уменьшено) значение bsf либо не будут просмотрены все кандидаты без улучшения значения bsf. Если значение bsf улучшено, алгоритм переходит к началу, вычисляя битовую карту и пытаясь отбросить как бесперспективные оставшиеся подпоследовательности.

По окончании работы алгоритма в элементах профиля расстояния записано значение DTW-расстояния между запросом и соответствующей подпоследовательностью либо зна-

чение $+\infty$, которое означает, что данная подпоследовательность заведомо не похожа на запрос, причем в этом случае значение расстояния не вычислялось.

Алг. 4 SCORING(IN: $DistProfile^1, \ldots, DistProfile^d$, k; OUT: $kNNset$)

1: $Rating \leftarrow \overline{0}$; $s \leftarrow 0$; $kNNset \leftarrow \varnothing$
2: #pragma omp parallel for
3: **for** $i \in 1..N$ **do**
4: **for** $j \in 1..d$ **do**
5: $Rating(i) \leftarrow Rating(i) + \frac{1}{DistProfile^j(i)+\varepsilon} \cdot \frac{d-j+1}{d}$
6: **while** TRUE **do**
7: $s \leftarrow s + 1$
8: $maxRating \leftarrow \max\limits_{i \in 1..N} Rating(i)$; $maxIdx \leftarrow \arg\max\limits_{i \in 1..N} Rating(i)$
9: **if** $|maxIdx - prevIdx| > m$ **then**
10: $kNNset \leftarrow kNNset \cup \{T_{maxIdx,\,m}\}$; $prevIdx \leftarrow maxIdx$
11: **else**
12: $Rating(maxIdx) \leftarrow -\infty$
13: **if** $|kNNset| = k$ **or** $s = N$ **then**
14: **break**
15: **return** $kNNset$

Алг. 4 показывает псевдокод шага скоринга, на котором выполняется отбор подпоследовательностей исходного ряда для восстановления пропущенного значения. Сначала на основе профилей расстояния всех опорных рядов, вычисленных на предыдущем шаге, параллельно вычисляется рейтинг интервалов (строки 1–5). Вычисления выполняются с помощью двух вложенных циклов: внешний — по интервалам, внутренний — по опорным рядам; внешний цикл распараллеливается. После этого в векторе $Rating$ мы находим top-k наибольших элементов так, чтобы их индексы различались не менее чем на m (строки 6–14).

Рейтинг подпоследовательности $T_{i,\,m}$ определяется следующим образом:

$$Rating(T_{i,\,m}) = \sum_{s=1}^{d} \frac{w(s,i)}{DistProfile^s(i) + \varepsilon}, \tag{16}$$

где $DistProfile^s(i) = \mathrm{DTW}(Q^s, R^s_{i,\,m})$ — это DTW-расстояние между запросом s-го опорного ряда и его i-й подпоследовательностью, ε — машинный эпсилон (верхняя граница относительной ошибки из-за округления в арифметике с плавающей точкой), $w : \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ — весовая функция.

Вес подпоследовательности в формуле (16) позволяет учитывать корреляцию между восстанавливаемым и опорным временными рядами, а также индекс подпоследовательности. Например, весовая функция позволяет выбирать для восстановления, соответственно, подпоследовательности из более коррелированных опорных временных рядов и из наиболее недавних моментов времени. В нашем исследовании мы берем весовую функцию $w(s,i) = \frac{d-s+1}{d}$, которая имеет следующую семантику: больший вес имеет подпоследовательность более коррелированного временного ряда, индекс подпоследовательности не учитывается.

В итоге реконструкция пропущенного значения t_n выполняется следующим образом:

$$\tilde{t}_n = \frac{1}{k} \sum_{T_{i,\,m} \in kNNset} t_m. \tag{17}$$

4. Вычислительные эксперименты

Для оценки эффективности разработанного алгоритма нами были проведены вычислительные эксперименты, в которых исследовались точность восстановления и производительность алгоритма.

Для оценки *точности восстановления* пропущенных значений нами используется мера среднеквадратичной ошибки *RMSE (Root Mean Square Error)*, определяемая следующим образом:

$$RMSE = \sqrt{\frac{1}{h} \sum_{i=1}^{h} (t_i - \tilde{t}_i)^2}, \tag{18}$$

где t_i — фактическое значение элемента ряда, считающегося пропущенным, $\tilde{t}_i$ — восстановленное значение, h — количество пропущенных элементов временного ряда.

Производительность алгоритма понимается как среднее время, затрачиваемое на восстановление одного пропущенного элемента временного ряда. В экспериментах запуск программы осуществлялся 10 раз и в качестве итогового времени вычисления использовалось медианное значение. Эксперименты были проведены на оборудовании Лаборатории суперкомпьютерного моделирования ЮУрГУ [23]: процессор Intel Xeon E5-2687W v2 (8 ядер @3.40 GHz).

Для исследований нами использовался фреймворк ORBITS [24], который реализует одноименный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени и обеспечивает проведение экспериментов со следующими алгоритмами аналогичного назначения сторонних авторов: OGDImpute [25], SPIRIT [26], SAGE [27], TKCM [15].

Таблица 1. Наборы данных

Набор данных	Количество рядов	Длина	Предметная область
BAFU	10	50 000	Сброс воды в реках Швейцарии
Chlorine	50	1 000	Распространение хлора в системе распределения питьевой воды
Climate	10	5 000	Погода в различных локациях Северной Америки
MADRID	10	25 000	Трафик на дорогах Мадрида
MAREL	10	50 000	Характеристики морской воды в Ла-Манше

В экспериментах нами использовались наборы данных, представленные в табл. 1. Chlorine [28] представляет собой набор синтетических временных рядов, который создан путем моделирования системы распределения питьевой воды и описывает концентрацию хлора. Распространение хлора в системе вызывает фазовые сдвиги в рядах набора данных. BAFU [29] представляет собой набор временных рядов с данными о сбросе воды в 10 реках Швейцарии. Частота измерений высока для отслеживания быстро меняющегося давления воды в сезон дождей. Набор содержит периодические временные ряды, некоторые

из которых смещены во времени. Набор Climate [30] представляет ежемесячные агрегированные климатические данные, собранные с метеостанций в различных локациях Северной Америки в 1990–2002 гг. Временные ряды нерегулярны и содержат спорадические всплески. Набор MADRID [31] содержит данные автоматических регистраторов дорожного движения, установленных на городских дорогах и автострадах Мадрида. Набор MAREL [32] содержит данные о различных химических и биологических характеристиках морской воды в проливе Ла-Манш.

В экспериментах для каждого набора данных 10% последних точек одного из рядов, идущие подряд, подлежали восстановлению. При этом осуществлялось накопление ошибки: для восстановления каждого элемента использовались ранее восстановленные, а не исходные значения всех предыдущих элементов. При восстановлении рассматривалось два случая относительно количества опорных рядов: минимальное (число, необходимое для запуска эксперимента в фреймворке ORBITS) и максимальное (общее число рядов в наборе за исключением восстанавливаемого ряда). В экспериментах с алгоритмами-аналогами устанавливались такие параметры их запуска, которые рекомендованы разработчиками фреймворка ORBITS для обеспечения наивысшей точности восстановления [24]. Для разработанного алгоритма во всех экспериментах использовались следующие параметры: длина запроса $m = 50$, ограничение полосы Сако—Чиба $r = \lceil 0.25m \rceil$, количество соседей $k = 3$.

Таблица 2. Результаты экспериментов

Набор данных	К-во опор. рядов, d	Критерий сравнения	SAGE	OGD-Impute	ORBITS		SPIRIT	TKCM	Наш алгоритм
					$k = 2$	$k = 3$			
BAFU	3	**Точность**, RMSE	14.82	1.18	10.03	10.09	0.66	0.18	**0.16**
		Производительность, мс	0.49	1.23	0.09	0.20	*0.003*	99.00	31.86
	9	**Точность**, RMSE	2.77	1.18	9.92	10.12	**0.07**	0.18	0.19
		Производительность, мс	0.50	3.09	0.18	0.25	*0.003*	99.00	93.8
Chlorine	4	**Точность**, RMSE	1.05	0.59	0.70	0.75	0.43	0.41	**0.03**
		Производительность, мс	0.05	0.40	0.14	0.19	*0.003*	2.00	0.80
	49	**Точность**, RMSE	0.29	0.59	0.21	0.08	0.10	0.41	**0.02**
		Производительность, мс	0.03	2.58	0.12	1.16	*0.02*	2.00	11.90
Climate	3	**Точность**, RMSE	3 888.78	33.29	366.78	366.74	4.32	10.19	**1.17**
		Производительность, мс	0.07	0.28	0.09	0.13	*0.007*	9.94	3.02
	9	**Точность**, RMSE	1 587.31	33.29	28.62	6.49	6.13	10.19	**1.51**
		Производительность, мс	0.08	0.71	0.07	0.04	*0.004*	9.97	13.7
MADRID	3	**Точность**, RMSE	903.48	1.09e+12	396.40	845.52	1 090.35	536.14	**212.97**
		Производительность, мс	0.29	0.71	0.10	0.18	*0.003*	47.79	14.67
	9	**Точность**, RMSE	739.84	1.09e+12	249.74	645.32	381.20	536.14	**202.42**
		Производительность, мс	0.32	1.79	0.05	0.32	*0.004*	48.09	42.22
MAREL	3	**Точность**, RMSE	37.86	2.78	10.82	11.18	11.29	**2.75**	4.60
		Производительность, мс	0.56	1.17	0.21	0.30	*0.003*	101.08	33.16
	9	**Точность**, RMSE	81.29	2.78	6.05	4.69	4.63	**2.75**	4.71
		Производительность, мс	0.48	2.95	0.02	0.02	*0.004*	95.90	92.80

Итоги экспериментов представлены в табл. 2, где курсивом выделены результаты наиболее быстрого алгоритма, полужирным — наиболее точного. Можно видеть, что предложенный параллельный алгоритм показывает, как правило, наиболее высокую точность восстановления, уступая лишь алгоритмам TKCM и SPIRIT соответственно на наборах данных MAREL и BAFU (в случае максимального количества опорных рядов). Мы можем заключить, что применение эвристики о схожем поведении восстанавливаемого и опорных временных рядов вкупе с мерой DTW, хорошо определяющей схожесть по форме подпоследовательности ряда, позволило предложенному нами алгоритму достичь высокой точности

восстановления. Однако наиболее производительным является последовательный алгоритм, SPIRIT, что объясняется следующим образом. Применение параллельных вычислений и отбрасывания бесперспективных кандидатов с помощью техники нижних границ позволило нашему алгоритму существенно ускорить время восстановления, однако заложенные в его природе накладные расходы на вычисление меры DTW, как правило, все же не могут быть сведены к нулю.

Отметим также, что в экспериментах предложенный алгоритм показывает быстродействие восстановления в диапазоне 0.8–93 мс в лучшем и худшем случаях соответственно. Данный результат допускает применение разработанного алгоритма в режиме реального времени. Например, в статье [33] указывается, что в системах автоматизации управления для сети передачи данных, обслуживающей датчики измерения температуры, влажности, давления цикл передачи данных составляет до 100 мс. В каталоге температурных датчиков компании Emerson [34] 2021 г., одного из ведущих мировых производителей измерительных систем, указывается, что беспроводные температурные датчики имеют период обновления данных не менее 1 с.

Заключение

В работе затронута проблема восстановления пропущенных значений в потоковом временном ряде в режиме реального времени, которая встречается в настоящее время в приложениях Интернета вещей, персональной медицины, цифровая индустрии и др. Пропуски в данных временного ряда, возникающие в таких приложениях ввиду технических сбоев или человеческого фактора, как правило, неприемлемы и подлежат замене на правдоподобные значения, синтезированные на лету.

В статье предложен новый параллельный алгоритм для многоядерного процессора, который выполняет восстановление пропущенного значения потокового временного ряда в режиме реального времени. Параллелизм вычислений реализован с помощью технологии программирования OpenMP. Алгоритм работает в предположении, что имеется набор опорных временных рядов, которые имеют семантическую связь с исходным рядом. Алгоритм применяет следующую эвристику: если в опорных рядах имеют место повторяющиеся (схожие) подпоследовательности, то в ряде, содержащем пропущенное значение, повторяющиеся подпоследовательности возникают в тех же временны́х интервалах.

Предложенный алгоритм выполняет следующие шаги: поиск по образцу, скоринг и реконструкция. Для каждого опорного ряда образцами поиска полагаются подпоследовательности заданной длины, которые оканчиваются в момент пропуска значения в исходном ряде. Далее для каждого опорного ряда выполняется поиск подпоследовательности, которая является самой похожей на образец в смысле меры DTW (Dynamic Time Warping) [5]. Мера DTW имеет квадратичную вычислительную сложность относительно длины подпоследовательности и определяется рекуррентными формулами, но в общем случае отражает схожесть формы образца поиска и подпоследовательности ряда лучше, чем евклидова метрика. Для ускорения данного шага используется техника нижних границ (lower bounding) [6], которые представляют собой меры схожести с вычислительной сложностью меньше, чем у меры DTW. Данная техника позволяет отбрасывать без вычисления DTW подпоследовательности, заведомо непохожие на образец поиска. Для каждого опорного ряда выполняется параллельное вычисление нижних границ, а также параллельно вычисляются DTW-

расстояния от образца до каждой подпоследовательности опорного ряда. Расстояния до отброшенных подпоследовательностей полагаются равными $+\infty$.

На шаге скоринга алгоритм, используя вычисленные ранее DTW-расстояния, определяет множество подпоследовательностей исходного ряда, применяемых для восстановления пропущенного значения. Данное множество состоит из k не пересекающихся друг с другом подпоследовательностей, которые являются наиболее значимыми для восстановления пропущенного значения. Значимость подпоследовательности определяется нами как функция от DTW-расстояний соответствующих подпоследовательностей в опорных рядах, подсчитанных на предыдущем шаге. Вычисление значимости всех подпоследовательностей исходного ряда выполняется параллельно. На финальном шаге реконструкции восстановленное значение вычисляется как среднее арифметическое последних элементов подпоследовательностей, найденных на шаге скоринга.

В вычислительных экспериментах с реальными и синтетическими данными предложенный алгоритм продемонстрировал точность восстановления, как правило, более высокую, чем у аналогов, и быстродействие восстановления, приемлемое для применения алгоритма в режиме реального времени.

В качестве возможного направления продолжения исследований мы рассматриваем разработку версии алгоритма для графического процессора.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 20-07-00140) и Министерства науки и высшего образования РФ (государственные задания FENU-2020-0022, FWNF-2022-0016).

Литература

1. Цымблер М.Л., Краева Я.А., Латыпова Е.А. и др. Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 16–36. DOI: 10.14529/cmse210302.
2. Иванов С.А., Никольская К.Ю., Радченко Г.И. и др. Концепция построения цифрового двойника города // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 4. С. 5–23. DOI: 10.14529/cmse200401.
3. Епишев В.В., Исаев А.П., Миниахметов Р.М. и др. Система интеллектуального анализа данных физиологических исследований в спорте высших достижений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. Т. 2, № 1. С. 44–54. DOI: 10.14529/cmse130105.
4. Абдуллаев С.М., Ленская О.Ю., Гаязова А.О. и др. Алгоритмы краткосрочного прогноза с использованием радиолокационных данных: оценка трасляции и композиционный дисплей жизненного цикла // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. Т. 3, № 1. С. 17–32. DOI: 10.14529/cmse140102.
5. Berndt D.J., Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series // Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03 / ed. by U.M. Fayyad, R. Uthurusamy. 1994. P. 359–370.

6. Ding H., Trajcevski G., Scheuermann P., *et al.* Querying and mining of time series data: experimental comparison of representations and distance measures // Proc. VLDB Endow. 2008. Vol. 1, no. 2. P. 1542–1552. DOI: 10.14778/1454159.1454226.
7. Цымблер М.Л., Полуянов А.Н. Параллельный алгоритм восстановления пропущенных значений потокового временного ряда в режиме реального времени // Параллельные вычислительные технологии – XVI международная конференция, ПаВТ'2022, Дубна, 29-31 марта 2022. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ. 2022. С. 128–140. DOI: 10.14529/pct2022.
8. Khayati M., Lerner A., Tymchenko Z., Cudré-Mauroux P. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series // Proc. VLDB Endow. 2020. Vol. 13, no. 5. P. 768–782. DOI: 10.14778/3377369.3377383.
9. Batista G.E.A.P.A., Monard M.C. An Analysis of Four Missing Data Treatment Methods for Supervised Learning // Appl. Artif. Intell. 2003. Vol. 17, no. 5-6. P. 519–533. DOI: 10.1080/713827181.
10. Troyanskaya O.G., Cantor M.N., Sherlock G., *et al.* Missing value estimation methods for DNA microarrays // Bioinform. 2001. Vol. 17, no. 6. P. 520–525. DOI: 10.1093/bioinformatics/17.6.520.
11. Hsu H., Yang A.C., Lu M. KNN-DTW Based Missing Value Imputation for Microarray Time Series Data // J. Comput. 2011. Vol. 6, no. 3. P. 418–425. DOI: 10.4304/jcp.6.3.418-425.
12. Phan T., Poisson É.C., Bigand A., Lefebvre A. DTW-Approach for uncorrelated multivariate time series imputation // 27th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2017, Tokyo, Japan, September 25-28, 2017 / ed. by N. Ueda, S. Watanabe, T. Matsui, *et al.* 2017. P. 1–6. DOI: 10.1109/MLSP.2017.8168165.
13. Phan T., Caillault É.P., Lefebvre A., Bigand A. Dynamic time warping-based imputation for univariate time series data // Pattern Recognit. Lett. 2020. Vol. 139. P. 139–147. DOI: 10.1016/j.patrec.2017.08.019.
14. Keogh E.J., Pazzani M.J. Derivative Dynamic Time Warping // Proceedings of the 1st SIAM International Conference on Data Mining, SDM 2001, Chicago, IL, USA, April 5-7, 2001 / ed. by V. Kumar, R.L. Grossman. 2001. P. 1–11. DOI: 10.1137/1.9781611972719.1.
15. Wellenzohn K., Böhlen M.H., Dignös A., *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series // Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017 / ed. by V. Markl, S. Orlando, B. Mitschang, *et al.* 2017. P. 330–341. DOI: 10.5441/002/edbt.2017.30.
16. Rakthanmanon T., Campana B.J.L., Mueen A., *et al.* Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping // ACM Trans. Knowl. Discov. Data. 2013. Vol. 7, no. 3. 10:1–10:31. DOI: 10.1145/2500489.
17. Краева Я.А., Цымблер М.Л. Совместное использование технологий MPI и OpenMP для параллельного поиска похожих подпоследовательностей в сверхбольших временных рядах на вычислительном кластере с узлами на базе многоядерных процессоров Intel Xeon Phi Knights Landing // Вычислительные методы и программирование. 2019. Т. 20, № 1. С. 29–44. DOI: 10.26089/NumMet.v20r104.

18. Kim S., Park S., Chu W.W. An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases // Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany / ed. by D. Georgakopoulos, A. Buchmann. 2001. P. 607–614. DOI: 10.1109/ICDE.2001.914875.

19. Fu A.W., Keogh E.J., Lau L.Y.H., Ratanamahatana C. Scaling and Time Warping in Time Series Querying // Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005 / ed. by K. Böhm, C.S. Jensen, L.M. Haas, *et al.* 2005. P. 649–660. URL: http://www.vldb.org/archives/website/2005/program/paper/thu/p649-fu.pdf.

20. Supinski B.R. de, Scogland T.R.W., Duran A., *et al.* The Ongoing Evolution of OpenMP // Proc. IEEE. 2018. Vol. 106, no. 11. P. 2004–2019. DOI: 10.1109/JPROC.2018.2853600.

21. Dau H.A., Keogh E., Kamgar K. и др. The UCR Time Series Classification Archive. 2018. URL: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (дата обращения: 12.04.2022).

22. Mueen A., Keogh E.J., Zhu Q., *et al.* Exact Discovery of Time Series Motifs // Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA. SIAM, 2009. P. 473–484. DOI: 10.1137/1.9781611972795.41.

23. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC resources of South Ural State University // 16th International Conference on Parallel Computational Technologies, PCT 2022, Dubna, Russia, March 29-31, 2022, Revised Selected Papers. Communications in Computer and Information Science. Vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

24. Khayati M., Arous I., Tymchenko Z., Cudré-Mauroux P. ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams // Proc. VLDB Endow. 2020. Vol. 14, no. 3. P. 294–306. DOI: 10.5555/3430915.3442429.

25. Anava O., Hazan E., Zeevi A. Online Time Series Prediction with Missing Data // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11, 2015. 2015. P. 2191–2199. URL: http://proceedings.mlr.press/v37/anava15.html.

26. Papadimitriou S., Sun J., Faloutsos C. Streaming Pattern Discovery in Multiple Time-Series // Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005 / ed. by K. Böhm, C.S. Jensen, L.M. Haas, *et al.* 2005. P. 697–708. DOI: 10.5555/1083592.1083674.

27. Balzano L., Chi Y., Lu Y.M. Streaming PCA and Subspace Tracking: The Missing Data Case // Proc. IEEE. 2018. Vol. 106, no. 8. P. 1293–1310. DOI: 10.1109/JPROC.2018.2847041.

28. Chlorine Dataset. URL: https://www.cs.cmu.edu/afs/cs/project/spirit-1/www/ (дата обращения: 03.09.2021).

29. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. URL: https://www.hydrodaten.admin.ch/en (дата обращения: 03.09.2021).

30. Lozano A.C., Li H., Niculescu-Mizil A., *et al.* Spatial-temporal causal modeling for climate change attribution // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F. Elder IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

31. Laña I., Olabarrieta I., Vélez M., Del Ser J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques // Transportation Research Part C: Emerging Technologies. 2018. Vol. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

32. Lefebvre A. MAREL Carnot data and metadata from Coriolis Data Centre. SEANOE. 2015. DOI: 10.17882/39754.

33. Лопухов И. Сети Real-Time Ethernet: от теории к практической реализации // СТА: Современные технологии автоматизациии. 2010. Т. 10, № 3. С. 8–15.

34. Каталог 2021. Датчики температуры Emerson. URL: https://www.c-o-k.ru/library/catalogs/emerson/110477.pdf (дата обращения: 03.09.2021).

Цымблер Михаил Леонидович, д.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Полуянов Андрей Николаевич, к.т.н., старший научный сотрудник, Институт математики им. С.Л. Соболева СО РАН (Омск, Российская Федерация)

Краева Яна Александровна, старший преподаватель, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse220305

PARALLEL ALGORITHM FOR REAL-TIME SENSOR DATA RECOVERY FOR A MANY-CORE PROCESSOR

© 2022 M.L. Zymbler[1], A.N. Poluyanov[2], Ya.A. Kraeva[1]

[1]*South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia),*

[2]*S.L. Sobolev Institute of Mathematics SB RAS (Pevtsova str. 13, Omsk, 644043 Russia)*

E-mail: mzym@susu.ru, andrey.poluyanov@gmail.com, kraevaya@susu.ru

Received: 30.07.2022

Currently, in many subject areas, the processing of sensor data in real time assumes imputation of values missed due to a technical failure or a human factor. This article proposes a parallel algorithm for imputation the missing values of a streaming time series in real time for a many-core processor. The algorithm employs a set of reference time series that have a semantic relationship with the original time series. The algorithm exploits the following heuristics: if there are repeated (similar) subsequences in the reference time series, then in the time series containing the missing value, repeated subsequences occur in the same time intervals. For each reference time series, a query is defined as a subsequence of a given length ending at the moment when the value in the original time series was missed. The similarity of the subsequences with the query is determined based on the DTW (Dynamic Time Warping) measure that is of quadratic computational complexity relative to the subsequence length. The algorithm employs the lower bounding technique to discard subsequences that are obviously dissimilar to the query, without calculating DTW. The lower bounds have less complexity than DTW and are calculated in parallel. The imputed value is calculated as the arithmetic mean of the last elements of the found intervals. In computational experiments, the proposed algorithm demonstrates high imputation accuracy in comparison with analogs and performance acceptable for real-time applications.

Keywords: time series, imputation of missing values, parallel algorithm, many-core CPU, DTW, lower bounding.

FOR CITATION

Zymbler M.L., Poluyanov A.N., Kraeva Ya.A. Parallel Algorithm for Real-time Sensor Data Recovery for a Many-core Processor. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 3. P. 69–90. (in Russian) DOI: 10.14529/cmse220305.

References

1. Zymbler M.L., Kraeva Y.A., Latypova E.A., *et al.* Cleaning Sensor Data in Intelligent Heating Control System. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 3. P. 16–36. (in Russian) DOI: 10.14529/cmse210302.
2. Ivanov S.A., Nikolskaya K.Y., Radchenko G.I., *et al.* Digital Twin of a City: Concept Overview. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2020. Vol. 9, no. 4. P. 5–23. (in Russian) DOI: 10.14529/cmse200401.
3. Epishev V.V., Isaev A.P., Miniakhmetov R.M., *et al.* Physiological Data Mining System For Elite Sports. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2013. Vol. 2, no. 1. P. 44–54. (in Russian) DOI: 10.14529/cmse130105.
4. Abdoulaev S.M., Lenskaia O.U., Gayazova A.O., *et al.* Short-Range Forecasting Algorithms Using Radar Data: Translation Estimate And Life-Cycle Composite Display. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2014. Vol. 3, no. 1. P. 17–32. (in Russian) DOI: 10.14529/cmse140102.
5. Berndt D.J., Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series. Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03 / ed. by U.M. Fayyad, R. Uthurusamy. 1994. P. 359–370.
6. Ding H., Trajcevski G., Scheuermann P., *et al.* Querying and mining of time series data: experimental comparison of representations and distance measures. Proc. VLDB Endow. 2008. Vol. 1, no. 2. P. 1542–1552. DOI: 10.14778/1454159.1454226.
7. Zymbler M.L., Poluyanov A.N. Parallel algorithm for imputation of missing values in a streaming time series in real time. Parallel Computational Technologies – 16th International Conference, PCT 2022, Dubna, Russia, March 29-31, 2022. Short papers and posters. Chelyabinsk: SUSU Publishing Center. 2022. P. 128–140. (in Russian) DOI: 10.14529/pct2022.
8. Khayati M., Lerner A., Tymchenko Z., Cudré-Mauroux P. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series. Proc. VLDB Endow. 2020. Vol. 13, no. 5. P. 768–782. DOI: 10.14778/3377369.3377383.

9. Batista G.E.A.P.A., Monard M.C. An Analysis of Four Missing Data Treatment Methods for Supervised Learning. Appl. Artif. Intell. 2003. Vol. 17, no. 5-6. P. 519–533. DOI: 10.1080/713827181.

10. Troyanskaya O.G., Cantor M.N., Sherlock G., *et al.* Missing value estimation methods for DNA microarrays. Bioinform. 2001. Vol. 17, no. 6. P. 520–525. DOI: 10.1093/bioinformatics/17.6.520.

11. Hsu H., Yang A.C., Lu M. KNN-DTW Based Missing Value Imputation for Microarray Time Series Data. J. Comput. 2011. Vol. 6, no. 3. P. 418–425. DOI: 10.4304/jcp.6.3.418-425.

12. Phan T., Poisson É.C., Bigand A., Lefebvre A. DTW-Approach for uncorrelated multivariate time series imputation. 27th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2017, Tokyo, Japan, September 25-28, 2017 / ed. by N. Ueda, S. Watanabe, T. Matsui, *et al.* 2017. P. 1–6. DOI: 10.1109/MLSP.2017.8168165.

13. Phan T., Caillault É.P., Lefebvre A., Bigand A. Dynamic time warping-based imputation for univariate time series data. Pattern Recognit. Lett. 2020. Vol. 139. P. 139–147. DOI: 10.1016/j.patrec.2017.08.019.

14. Keogh E.J., Pazzani M.J. Derivative Dynamic Time Warping. Proceedings of the 1st SIAM International Conference on Data Mining, SDM 2001, Chicago, IL, USA, April 5-7, 2001 / ed. by V. Kumar, R.L. Grossman. 2001. P. 1–11. DOI: 10.1137/1.9781611972719.1.

15. Wellenzohn K., Böhlen M.H., Dignös A., *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series. Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21-24, 2017 / ed. by V. Markl, S. Orlando, B. Mitschang, *et al.* 2017. P. 330–341. DOI: 10.5441/002/edbt.2017.30.

16. Rakthanmanon T., Campana B.J.L., Mueen A., *et al.* Addressing Big Data Time Series: Mining Trillions of Time Series Subsequences Under Dynamic Time Warping. ACM Trans. Knowl. Discov. Data. 2013. Vol. 7, no. 3. 10:1–10:31. DOI: 10.1145/2500489.

17. Kraeva Y.A., Zymbler M.L. The use of MPI and OpenMP technologies for subsequence similarity search in very long time series on a computer cluster system with nodes based on the Intel Xeon Phi Knights Landing many-core processor. Numerical Methods and Programming. 2019. Vol. 20, no. 1. P. 29–44. (in Russian) DOI: 10.26089/NumMet.v20r104.

18. Kim S., Park S., Chu W.W. An Index-Based Approach for Similarity Search Supporting Time Warping in Large Sequence Databases. Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany / ed. by D. Georgakopoulos, A. Buchmann. 2001. P. 607–614. DOI: 10.1109/ICDE.2001.914875.

19. Fu A.W., Keogh E.J., Lau L.Y.H., Ratanamahatana C. Scaling and Time Warping in Time Series Querying. Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005 / ed. by K. Böhm, C.S. Jensen, L.M. Haas, *et al.* 2005. P. 649–660. URL: http://www.vldb.org/archives/website/2005/program/paper/thu/p649-fu.pdf.

20. Supinski B.R. de, Scogland T.R.W., Duran A., *et al.* The Ongoing Evolution of OpenMP. Proc. IEEE. 2018. Vol. 106, no. 11. P. 2004–2019. DOI: 10.1109/JPROC.2018.2853600.

21. Dau H.A., Keogh E., Kamgar K., *et al.* The UCR Time Series Classification Archive. 2018. URL: https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (accessed: 12.04.2022).

22. Mueen A., Keogh E.J., Zhu Q., *et al.* Exact Discovery of Time Series Motifs. Proceedings of the SIAM International Conference on Data Mining, SDM 2009, April 30 - May 2, 2009, Sparks, Nevada, USA. SIAM, 2009. P. 473–484. DOI: 10.1137/1.9781611972795.41.

23. Dolganina N., Ivanova E., Bilenko R., Rekachinsky A. HPC resources of South Ural State University. 16th International Conference on Parallel Computational Technologies, PCT 2022, Dubna, Russia, March 29-31, 2022, Revised Selected Papers. Communications in Computer and Information Science. Vol. 1618 / ed. by L. Sokolinsky, M. Zymbler. Springer, 2022. P. 43–55. DOI: 10.1007/978-3-031-11623-0_4.

24. Khayati M., Arous I., Tymchenko Z., Cudré-Mauroux P. ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams. Proc. VLDB Endow. 2020. Vol. 14, no. 3. P. 294–306. DOI: 10.5555/3430915.3442429.

25. Anava O., Hazan E., Zeevi A. Online Time Series Prediction with Missing Data. Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, July 6-11, 2015. 2015. P. 2191–2199. URL: http://proceedings.mlr.press/v37/anava15.html.

26. Papadimitriou S., Sun J., Faloutsos C. Streaming Pattern Discovery in Multiple Time-Series. Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005 / ed. by K. Böhm, C.S. Jensen, L.M. Haas, *et al.* 2005. P. 697–708. DOI: 10.5555/1083592.1083674.

27. Balzano L., Chi Y., Lu Y.M. Streaming PCA and Subspace Tracking: The Missing Data Case. Proc. IEEE. 2018. Vol. 106, no. 8. P. 1293–1310. DOI: 10.1109/JPROC.2018.2847041.

28. Chlorine Dataset. URL: https://www.cs.cmu.edu/afs/cs/project/spirit-1/www/ (accessed: 03.09.2021).

29. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. URL: https://www.hydrodaten.admin.ch/en (accessed: 03.09.2021).

30. Lozano A.C., Li H., Niculescu-Mizil A., *et al.* Spatial-temporal causal modeling for climate change attribution. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F. Elder IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

31. Laña I., Olabarrieta I., Vélez M., Del Ser J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques. Transportation Research Part C: Emerging Technologies. 2018. Vol. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

32. Lefebvre A. MAREL Carnot data and metadata from Coriolis Data Centre. SEANOE. 2015. DOI: 10.17882/39754.

33. Lopukhov I. Real-Time Ethernet network: from theory to practical implementation. MAT: Modern automation technologies. 2010. Vol. 10, no. 3. P. 8–15.

34. Catalogue 2021. Emerson temperature sensors. URL: https://www.c-o-k.ru/library/catalogs/emerson/110477.pdf (accessed: 03.09.2021).

СВЕДЕНИЯ ОБ ИЗДАНИИ

Научный журнал «Вестник ЮУрГУ. Серия «Вычислительная математика и информатика» основан в 2012 году.

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

Главный редактор — Л.Б. Соколинский.

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

Журнал включен в Реферативный журнал и Базы данных ВИНИТИ; индексируется в библиографической базе данных РИНЦ. Журнал размещен в открытом доступе на Всероссийском математическом портале MathNet. Сведения о журнале ежегодно публикуются в международной справочной системе по периодическим и продолжающимся изданиям «Ulrich's Periodicals Directory».

Решением Президиума Высшей аттестационной комиссии Министерства образования и науки Российской Федерации журнал включен в «Перечень рецензируемых научных изданий, в которых должны быть опубликованы основные научные результаты на соискание ученой степени кандидата наук, на соискание ученой степени доктора наук» по научным специальностям и соответствующим им отраслям науки: 2.3.5 – Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей (физико-математические науки), 05.13.17 – Теоретические основы информатики (физико-математические науки).

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

Адрес редакции, издателя: 454080, г. Челябинск, проспект Ленина, 76, Издательский центр ЮУрГУ, каб. 32.

ПРАВИЛА ДЛЯ АВТОРОВ

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

   Россия 454080, г. Челябинск, пр. им. В.И. Ленина, 76, ЮУрГУ, кафедра СП,

   ответственному секретарю Цымблеру М.Л.
3. Адрес электронной почты редакции: `vestnikvmi@susu.ru`
4. **Плата с авторов за публикацию рукописей не взимается, и гонорары авторам не выплачиваются.**

ВЕСТНИК
ЮЖНО-УРАЛЬСКОГО
ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА
Серия
«ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»
Том 11, № 3
2022

16+

Техн. редактор *А.В. Миних*

Издательский центр Южно-Уральского государственного университета

Подписано в печать 26.09.2022. Дата выхода в свет 03.10.2022. Формат 60×84 1/8. Печать цифровая.
Усл. печ. л. 10,69. Тираж 500 экз. Заказ 355/307. Цена свободная.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, проспект Ленина, 76.