ISSN 2079-3316 **ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ** **т. 12**, № 1(48), с. 21–113

ББК 32.972.11:32.971.32-043
ГРНТИ 50.07.05
УДК 519.681.3+004.415.3

Б. Я. Штейнберг, О. Б. Штейнберг

Преобразования программ — фундаментальная основа создания оптимизирующих распараллеливающих компиляторов

Аннотация. В работе рассматриваются преобразования программ, приводящие к ускорению. Приводятся публикации о различных параллельных вычислительных архитектурах и инструментах разработки эффективных программ для них. Рассматривается сочетание распараллеливания и оптимизации доступа к модулям памяти разного уровня. Отмечается, что отставание автоматической оптимизации программ от потребностей новых архитектур сдерживает развитие новых перспективных вычислительных систем. Формулируются задачи развития теории преобразования программ и оптимизирующих (распараллеливающих) компиляторов, которые могли бы привести к существенному повышению производительности труда программистов. Статья обосновывает призыв к модернизации оптимизирующей компиляции.

Ключевые слова и фразы: оптимизирующий компилятор, параллельные вычисления, преобразования программ, локальность данных, оптимизация обращений к памяти, тайл.

Посвящается памяти Владимира Анатольевича Евстигнеева, который начиная с 60-ых годов тщательно собирал библиографию по оптимизации, векторизации и распараллеливанию программ.

Содержание

Исследование выполнено при финансовой поддержке РФФИ («Экспансия») в рамках научного проекта № 19-17-50249/19.

Введение

Цель данной статьи — на основе обозреваемых публикаций определить возможные направления развития теории преобразования программ и особенно оптимизирующей (в т.ч. распараллеливающей) компиляции. Обзор достигнутого приводится для формулировки задач будущих исследований. Детали теории преобразования программ можно найти в приводимых ссылках на книги и статьи, включая обзорные. В данной статье рассматриваются, в основном, преобразования интересные с точки зрения вычислительных систем последнего времени.

Преобразования программ — это изменения в программах, улучшающие какие-то их качества: быстродействие, читабельность, структурированность, надежность, переносимость и пр. В данной обзорной статье рассматриваются преобразования программ, направленные на повышение их быстродействия. Оптимизирующие преобразования программ используются как в отдельных прикладных программах, так и в программных инструментах: компиляторах, библиотеках, решателях пакетов прикладных программ.

Оптимизирующие преобразования программ можно классифицировать по разным признакам: машинно-ориентированные и машинно-независимые; языково-независимые и специфичные для языка, глобальные и локальные; автоматические и «ручные», низкоуровневые и высокоуровневые, преобразования с одного уровня языка в другой.

Растет многообразие вычислительных архитектур и их сложность, что создает проблему переносимости программ с сохранением высокой эффективности. Небольшая часть программ (например, пакет LAPACK для задач линейной алгебры) переносится с сохранением высокой

эффективности благодаря библиотекам (BLAS), которые разрабатываются для каждого процессора. Но множество приложений быстро расширяется, и проблема эффективной переносимости для них может быть решена с помощью семейства оптимизирующих компиляторов, которые должны трансформировать код к виду, хорошо отображаемому на архитектуру вычислительной системы. Отсутствие эффективных компиляторов сдерживает как развитие приложений, так и развитие новых вычислительных систем (например, высокопроизводительных процессоров с программируемой архитектурой и многоядерных систем на кристалле с распределенной локальной памятью таких, как Tile64, Epiphany или 1879ВМ8Я от НТЦ «Модуль»).

Развитие вычислительной техники ставит перед теорией преобразования программ новые задачи. Но развитие новых методов и инструментов разработки программ стимулируется и потребностями создания новых приложений: обработка и распознавание изображений; беспилотный транспорт; анализ больших графов (коммуникационных сетей); моделирование поведения сложных систем; исследования в микробиологии и биоинформатике; искусственный интеллект; криптография. При этом и многие старые задачи высокопроизводительных вычислений [1] сохраняют свою актуальность, например, NP-трудные [2].

Первый и очень полный (для своего времени) список литературы по оптимизирующим преобразованиям программ и компиляторам собирал Владимир Анатольевич Евстигнеев. Этот список литературы приводится в качестве приложения к данной статье[1], как памятник замечательному ученому и человеку.

1. Разнообразие параллельных вычислительных архитектур

В представленной статье данный раздел нужен постольку, поскольку оптимизирующие преобразования программ ориентируются, в конечном счете, на эффективное отображение на целевую вычислительную архитектуру.

Опишем основные классы вычислительных систем (ВС) и их модулей памяти. Выделяемые типы допускают многообразие комбинаций (гибридные архитектуры).

[1] http://psta.psiras.ru/read/psta2021_1_21-113-appendix.pdf.

Обычно рассматривают основные типы параллельных вычислительных архитектур: SIMD, MIMD, Pipeline. Такая классификация подразумевает отождествление высокой производительности и параллельности вычислений. Но высокая производительность существенно зависит и от структуры памяти ВС.

Основные типы памяти: память общая или распределенная.

Общая память. Общая память может иметь специфику доступа: многоканальная память, ортогональная память, ассоциативная память, иерархия модулей памяти, основная или вспомогательная (более быстрая, но с меньшим объемом). SIMD с общей памятью — векторная архитектура. MIMD с общей памятью — например, несколько процессоров на общей шине. MIMD с общей памятью — VLIW (суперскаляр).

Часто используется вспомогательная память: кэш-память или адресуемая локальная общая память. Регистровая память имеет существенное отличие от кэш-памяти: данные, записываемые в кэш-память, записываются и в оперативную память, а данные, лежащие в регистре не обязаны быть в оперативной памяти. Таким образом, операции чтения из кэш-памяти, в отличие от записи, не приводят к обращениям к оперативной памяти. Запись в регистры или в локальную адресуемую память, не приводят к обращениям к оперативной памяти.

Распределенная память. Способы коммуникаций: коммутатор или коммуникационная сеть. Коммуникационные сети: звезда, дерево, шина, кольцо, решетка (mesh), многокольцевая сеть, гиперкуб, однородный граф малого диаметра. Следует отметить, как вариант распределенной памяти: адресуемая локальная распределенная память на одном кристалле.

1.1. Вычислительные архитектуры более чем 10-летней давности

Архитектура SIMD (Single Instruction Multiple Data) допускает выполнение одновременно несколько одинаковых операций над различными данными. Примерами компьютеров с такой архитектурой являются ILLIAC-4, Connection Machine 2 (64000 процессоров) и отечественный суперкомпьютер ПС-2000 [3] и др. [4]. Несмотря на жесткость ограничений, на эту архитектуру эффективно отображается достаточно представительный класс алгоритмов, имеющих регулярную структуру: задачи линейной алгебры с заполненными матрицами, сеточные методы

решения уравнений математической физики в прямоугольных областях, быстрое преобразование Фурье и некоторые другие [5]. Архитектура SIMD синхронная, поскольку все процессоры в каждый момент выполняют одинаковые операции. К классу архитектур SIMD можно отнести и векторные архитектуры (пример векторного компьютера — Cray1).

MIMD (Many Instruction Multiple Data) - много команд — много данных. Мультитранспьютерные системы, массово-параллельные суперкомпьютеры с распределенной памятью МВС-100 и МВС-1000 [6–8] и подобные им кластерные системы [9–11]. Каждый процессор суперкомпьютера такой архитектуры может работать по своей индивидуальной программе. Возникают (решаемые) проблемы с синхронизацией работы процессоров, которая необходима при обменах данными. Как правило, каждый процессор имеет свою локальную память. Распределенные вычисления GRID на глобальных сетях тоже можно отнести к этому классу, хотя и с особой спецификой [12–14].

Векторные компьютеры ориентированы на эффективные операции над векторами. К векторным компьютерам можно отнести Cray1, ПС-3000, суперкомпьютеры японской фирмы NEC[2]. Векторные компьютеры, как правило, имеют общую память. Эти компьютеры синхронные и допускают эффективное автоматическое распараллеливание (векторизацию) для достаточно представительного класса программ. Поддержка векторных операций реализована на некоторых массовых универсальных процессорах и графических картах.

Вычислительный конвейер (PIPELINE) — это такая совокупность устройств, у которых выходы одних связаны непосредственно с входами других [15, 16]. Конвейерное соединение устройств позволяет экономить время на обращениях к памяти. Конвейеры могут использоваться на разных уровнях вычислений. Например, побитовый конвейер может вычислять произведение двух чисел. А суперкомпьютер ПС-2100 допускал режим работы, при котором в конвейер соединялись десять 64-процессорных суперкомпьютерных модулей. Конвейер эффективен при большом потоке однотипных вычислений, равномерно загружающих все устройства конвейера. Мультиконвейерные архитектуры рассматривались в [17, 18]. Жесткое соединение устройств, входящих в конвейер, предполагает, что конвейер будет вычислять только одну функцию (или несколько функций с близкими алгоритмами). Это ограничивает применение конвейеров. Много работ посвящено эффективному отображению фрагментов программ на конвейерные системы, причем эти же

[2] URL https://www.nec.com/en/global/solutions/hpc/sx/architecture.html?

алгоритмы оказываются применимы и к программному конвейеру. Компьютеры VLIW (Very Long Instruction Word) — с очень длинным командным словом допускают одновременное выполнение набора различных простых команд. Простыми командами в данном случае могут быть арифметические операции, нормализация вещественных чисел с плавающей запятой, считывание данных из памяти и др. Такие компьютеры распараллеливают линейные участки программ. К таким архитектурам относился, например, суперкомпьютер «Эльбрус» [19–21]. Идеи VLIW вошли в архитектуры многих микропроцессоров.

Для обмена данными между вычислительными устройствами или для перераспределения данных используется некоторая коммуникационная среда. В [22] представлена топологическая классификация коммуникационных сетей для суперкомпьютеров с распределенной памятью: шины, решетки, схемы n-мерного куба, иерархические сети. Кольцевая шина, соединяющая процессоры, была реализована в ПС-2000 [5]. В суперкомпьютере ILLIAC-IV процессорные элементы находились в узлах двумерной решетки. Многокольцевые коммуникационные сети [23] топологически изоморфны решеткам и для них справедливы соответствующие расчеты. Коммутаторы к суперкомпьютерам пришли из средств связи [24–27]. Показано, что для полнодоступной коммутации N входов необходимо порядка $N \log(N)$ коммутационных элементов. Таким образом, при достаточно больших N система полнодоступной коммутации может оказаться дороже всех соединяемых устройств вместе взятых, особенно, если коммутировать требуется не битовые сигналы, а 32- или 64-разрядные слова. По этой причине, полнодоступная коммутация используется для коммутации не очень большого количества элементов. Есть ряд работ по поиску такой топологии, что при большом количестве процессорных элементов МВС и фиксированном количестве связей диаметр графа соединений минимален [28–31].

В симметрических архитектурах SMP несколько процессоров размещаются на общей памяти. Симметрические архитектуры асинхронны и используются в качестве серверов. Многоядерные процессоры — это процессоры, в которых на одном кристалле размещается несколько ядер, каждое из которых эквивалентно самостоятельному универсальному процессору предыдущих поколений. Ядра такого процессора работают асинхронно и используют общую память. И симметрические и многоядерные процессоры используют принципы многопоточного программирования [32].

Новые отечественные процессоры и основанные на них вычислительные комплексы разрабатывает группа компаний «Эльбрус» и, затем, МЦСТ [20, 33].

Систолические вычислительные системы [34] аппаратно реализуют вычислительный алгоритм, удовлетворяющий специальным требованиям. Граф алгоритма должен быть плоским и однородным, более точно, он должен отображаться в одну из 3 существующих плоских мозаик (состоящую из треугольников, или квадратов, или шестиугольников).

Систолические системы могут быть очень эффективны, но это не универсальные программируемые системы. Систолические массивы могут быть интересны для использования на ПЛИС, поскольку их функционирование подобно конвейеру, а планарность графа схемы удобна для коммутации при размещении на матрице микросхемы.

Процессоры цифровой обработки сигналов рассматриваются, например, в [35].

Более подробные описания некоторых параллельных вычислительных архитектур можно найти в [5, 16, 22, 36–45].

Следует отметить относительность понятия «суперкомпьютер»: то, что вчера было суперкомпьютером, сегодня реализуется как система на кристалле.

1.2. Вычислительные архитектуры последних лет и проекты новых компьютеров

Вычислительные архитектуры последних лет зачастую опираются на идеи прошлых архитектур, но с поправкой на прогресс микроэлектроники: то, что раньше было суперкомпьютером, теперь может стать системой на кристалле.

Выпускаемые в последнее время процессоры поддерживают по нескольку каналов памяти[3]. Многоканальная память позволяет получить ускорение, пропорциональное количеству процессоров, если используемые данные лежат в разных модулях памяти, к которым подключены каналы памяти. К сожалению, многоканальная память управляется процессором, и программистам не предоставлены инструменты влияния на это.

Многоканальная оперативная память DDR4 [46] имеет диапазон частот от 2133 МГц до 4400 МГц (против DDR3 с диапазоном частот 800-2933 МГц).

[3] См. https://ru.wikipedia.org/wiki/Многоканальная_архитектура_памяти.

В последнее время появляются вычислительные архитектуры с локальной адресуемой памятью на микросхеме процессора. Доступ к такой памяти более быстрый, чем к оперативной. Кэш память управляется процессором, а обсуждаемая локальная память — программистом. Всякое данное, лежащее в кэш-памяти, лежит и в оперативной памяти. В локальную память можно поместить данное, не помещая его в оперативную память. Чтение из кэш-памяти и из локальной памяти занимают одинаковое время. А вот запись в кэш-память сопряжена с записью в оперативную память. По этой причине локальная память имеет преимущество по сравнению с кэш-памятью для задач, в которых много операций записи в память. С другой стороны, программирование микросхем с локальной адресуемой памятью может оказаться более сложным (например, для графических карт NVidia к языку С или ФОРТРАН следует добавить CUDA).

Наверное, впервые, локальная адресуемая память появилась у процессоров IBM Cell [**47**]. Локальная адресуемая память, общая для всех ядер, работающих по одной команде в режиме SIMD, используется в графических картах NVidia и ускорителях Intel Xeon Phi.

В процессоре Tile64 (2007 г.) было 64 работающих асинхронно ядра, каждое из которых имело свою локальную память [?334]. Видимо, рекордное количество асинхронно работающих ядер, каждое из которых имеет свою локальную память, заявлено у процессора Epiphany — 1024 ядра [**48**]. Следует отметить и отечественную многоядерную систему на кристалле НТЦ «Модуль», ядра которой имеют свою адресуемую локальную память [**49**].

Для серверных приложений выпущен процессор IBM Power 9 [**50**], способный обрабатывать одновременно 96 потоков.

Для мобильных приложений используются процессоры семейства ARM [**51**], использующие векторные ускорители mali.

Процессор Kirin Huawei имеет две разной производительности (и энергопотребления) четверки процессорных ядер архитектуры ARM.

На процессорах архитектуры ARM создан самый мощный на данный момент суперкомпьютер.

Следует отметить хорошие описания принципов построения вычислительных архитектур в книге Д. Харрис и С. Харрис [**52**].

Проблемы и перспективы создания квантовых компьютеров описаны в [**53**], проект фотонного компьютера — в [**54**].

1.3. Выводы к главе

Многие вычислительные архитектуры прошлых лет не должны быть преданны забвению, поскольку возникают в системах на кристалле или ПЛИС[4]. Рост многообразия вычислительных архитектур создает проблемы переносимости ПО с сохранением эффективности. Под эффективностью ПО понимается доля пиковой производительности вычислительной системы.

2. Методы и инструменты разработки быстрых программ

2.1. Компиляторы

Компилятор — основной инструмент программиста и один из наиболее сложных программных продуктов, о чем говорится в популярной интернет-статье Евгения Зуева «Редкая профессия» [55].

В работе [56] В А. Вальковский приводит обзор разрабатывавшихся в СССР (экспериментальных) распараллеливающих компиляторов. Обзор рассматривал 18 таких экспериментальных компиляторов, самый большой из которых содержал 10000 строк (а второе место по объему — 5300 строк). Объем кода современных распараллеливающих компиляторов на 2 порядка больше.

Для векторного суперкомпьютера Cray-1 был разработан эффективный распараллеливающий компилятор PARAFRASE. В статье [57] отмечается, что существующие (на момент публикации) многопроцессорные среды не могут обеспечить желаемую производительность в широком спектре реальных приложений, главным образом из-за недостаточной точности информации об их зависимостях. Предлагается символьный анализ для распараллеливания. На основе предлагаемой методологии разработана структура символьного анализа для распараллеливающего компилятора Parafrase-2.

В ИПУ РАН разрабатывался векторный суперкомпьютер с общей памятью ПС-3100, для которого тоже был написан распараллеливающий компилятор [58, 59].

Рассмотрим некоторые популярные компиляторы:

---

[4]например, систолические, см. блог Ю. Панчула «Десятиклассница из Сибири хочет стать проектировщицей процессоров. Почему бы ей не сделать нейроускоритель на ПЛИС?» URL https://habr/com/ru/post/432378

- ICC — наиболее эффективный по результатам тестирований, но сопоставимый с GCC и LLVM, генераторы кода которых на процессоры Интел также поддерживаются этой компанией. Компилятор ICC входит в состав пакета инструментов Intel Parallel Studio[5].
- MS-Visual Studio — популярный компилятор, уступающий по производительности генерируемого кода компиляторам с открытым кодом GCC и LLVM.
- GCC (GNU Compiler Collection) — семейство компиляторов с открытым кодом. Много парсеров с различных языков и много генераторов кода на широкое семейство процессоров разных компаний. Оптимизирующие преобразования программ проводятся в регистровом промежуточном представлении RTL.
- LLVM (Low Level Virtual Machine) — семейство компиляторов с открытым кодом, подобно GCC. Лицензия позволяет использовать LLVM в коммерческих продуктах.
- Portland Group Incorporation. PGI-Compiler[6] на вход принимает программы языков C, C++, Fortran. Компиляторы PGI распараллеливают последовательные программы на многоядерные архитектуры и графические ускорители с помощью OpenACC, OpenMP, CUDA. PGI используется на суперкомпьютере Top 500#1 2019 Summit Supercomputer at Oak Ridge National Lab.
- МЦСТ — разработчик отечественных процессоров «Эльбрус» с полным комплектом системного программного обеспечения, включая оптимизирующий распараллеливающий компилятор.
- ROSE Compiler[7] является системой, состоящей из различных средств и инструментов. ROSE Tools может обрабатывать большие исходные коды C, C ++, Fortran, OpenMP и UPC, а также двоичные исполняемые файлы. ROSE – это инфраструктура компилятора с открытым исходным кодом и высокоуровневым внутренним представлением для создания и анализа source-to-source преобразований программ, применяемых к крупномасштабным C (C89 и C98), C ++ (C ++ 98 и C ++). 11), UPC, Fortran (77, 95, 2003), OpenMP, Java, Python, PHP и бинарным приложениям. На выходе выдается преобразованная программа на исходном языке

[5]пакет программных инструментов Intel Parallel Studio для разработки OpenMP параллельных программ, включает в себя компиляторы,библиотеки, анализаторы, профилировщик, URL https://software.intel.com/content/www/us/en/develop/tools/system-studio.html.

[6]URL https://www.pgroup.com/about/why-pgi.htm

[7]URL http://rosecompiler.org/

или бинарный код. ROSE Compiler содержит инструмент для автоматической расстановки прагм OpenMP в последовательный код написанный на C/C++.

Следует отметить, что давняя идея перенастраиваемого компилятора (retargetable compiler), который бы автоматизировал создание кода по формальному описанию целевой архитектуры, не нашла развития, в частности, из-за растущего многообразия архитектур.

2.2. Распараллеливающие системы

Отметим наиболее заметные распараллеливающие программные системы.

В [60] описана Т-система — среда параллельного программирования с поддержкой автоматического динамического распараллеливания программ. Программы пишутся на языке С с использованием специальной библиотеки функций, обеспечивающих интерфейс с ядром Т-системы.

В помощь пользователям параллельных суперкомпьютеров создавались различные сервисные программные системы [61–65].

Распараллеливающие системы — это компилирующие экспериментальные программные инструменты, ориентированные на преобразование последовательных программ к параллельные. Это не совсем полноценные компиляторы. На выходе, как и на входе, как правило, текст на высокоуровневом языке. Выходной текст может сопровождаться информацией о фрагментах кода, которые можно выполнять параллельно. В распараллеливающих системах бывают и экспериментальные генераторы кода.

В Новосибирске (ИСИ СО РАН) разрабатывалась система автоматического распараллеливания программ ПРОГРЕСС [66], в НИВЦ МГУ была реализована исследовательская система V-Ray, предназначенная для выявления возможностей параллелизма в последовательных ФОРТРАН программах [36].

Система исследования (тестирования) распараллеливающих и оптимизирующих компиляторов SUIF[8] ориентировалась на популярные параллельные архитектуры своего времени. Исследуются возможности отображения программ и на вычислительные системы с архитектурой VLIW [67], с распределенной памятью, и на Hyper-Threading процессоры. Наиболее интересны работы из этого проекта Моники Лэм [68–70].

[8]Stanford University Intermediate Format http:/suif.stanford.edu/research.

Распараллеливающая система POLARIS[9] ориентировалась на разработку распараллеливающих компиляторов с языков ФОРТРАН и С для параллельных систем с общей памятью (архитектура SMP). Система разрабатывалась в Urbana University с привлечением большого числа студентов и аспирантов под руководством D. Padua. Вероятно, одна из проблем этого большого проекта в том, что граф программных зависимостей строился не локально для преобразуемых фрагментов кода, а для всей программы в целом после каждого преобразования — это, конечно, большие затраты времени.

В [**71**] обсуждался проект системы F-Ray автоматического распараллеливания ФОРТРАН-программ на основе системы V-Ray для суперкомпьютеров с распределенной памятью. Этот проект ориентировался на минимизацию межпроцессорных пересылок. В [**72**] представлена инструментальная система поддержки распараллеливания программ, дополняющая систему V-Ray. В [**73**] формулируется абстрактное условие отсутствия обменов на основе матрицы информационных разрезов.

Система поддержки распараллеливания программ [**74**–**77**] должна была позволять пользователю выбирать преобразования из специальной библиотеки и применять их к исходной последовательной программе. При этом на пользователя ложится анализ эквивалентности преобразования, анализ целесообразности, анализ корректности и выбор преобразуемого фрагмента.

Преобразования программ разрабатываются в ОРС (Оптимизирующей распараллеливающей системе) [**78**–**100**].

В ИПМ им. Келдыша разработан язык сверхвысокого уровня для параллельного программирования задач математической физики НОРМА [**101**–**103**]. Сделаны компиляторы с этого языка в ФОРТРАН-77 распараллеливающий компилятор в ФОРТРАН-PVM. Язык НОРМА не допускает кратных присваиваний. Проект создавался для математиков совсем не знакомых с программированием, но умеющих писать на компьютере формулы.

В работах [**104**–**124**] идет речь о DVM-системе, в которой обычные языки программирования ФОРТРАН и Си снабжены дополнительными параллельными командами. Эти команды при компиляции на обычном трансляторе для ПК воспринимаются, как комментарии, и не исполняются, а при компиляции DVM-компилятором —

[9]Polaris Research Group,
https://sites.google.com/site/polarisresearchgroup/publications.

генерируют параллельный код. Аналогично устроено расширение системы OpenMP [125] для мультипроцессоров с общим адресным пространством. В настоящее время DVM-система позволяет разрабатывать параллельный код на языках Си и ФОРТРАН для широкого класса различных вычислительных архитектур, включая многоядерные процессоры, графические карты, высокопроизводительные кластеры и др. [106–125][10]. Методы и средства параллельного программирования представлены также в [126, 127].

Распараллеливающая система Pluto[11] и система анализа и преобразований пространства итераций гнезда циклов Polly[12] изучают гнезда циклов с точки зрения распараллеливания и локализации данных. В этих проектах предлагаются изменения обхода точек пространства итераций. Это красивые алгоритмы. Но такие алгоритмы иногда дают замедление при распараллеливании, а для ускорения требуется сочетание распараллеливания с предварительным применением тайлинга.

2.3. Библиотеки, пакеты и проблема перенесения программ с сохранением высокой эффективности

Проблемы переносимости программ с сохранением эффективности рассмотрены в [128]. Под эффективностью кода понимается доля пиковой производительности компьютера, на котором этот код работает. Параллельный переносимый язык OpenCL[13] позволяет переносить параллельные программы с одних параллельных вычислительных архитектур на другие, но, при этом, изменение эффективности может быть более, чем на порядок [128, 129]. Причины изменения эффективности при переносе программ, скорее всего, в различиях в структуре памяти в разных вычислительных системах.

Библиотеки программ содержат функции, часто используемые в некоторой предметной области. Высокопроизводительные библиотеки пишутся иногда с использованием ассемблерных функций и не переносятся с процессора на процессор или переносятся с потерями эффективности.

---

[10] см. также http://dvm-system.org/ru/about/

[11] URL http://pluto-compiler.sourceforge.net/

[12] URL https://www.sites.google.com/site/parallelizationforllvm/why-notpolly

[13] Arm Mali GPU OpenCL Version 3.11 Developer Guide URL https://static.docs.arm.com/100614/0311/arm_mali_gpu_opencl_developer_guide_100614_0311_00_en.pd

Библиотека BLAS (basic linear algebra subprograms) входит в состав системного программного обеспечения большинства процессоров общего назначения. Библиотечная программа умножения матриц (написанной по алгоритму, изучаемому на первом курсе мехмата) на современных процессорах может быть быстрее наивной студенческой программы на два порядка, но и объем кода библиотечной программы может быть больше объема кода наивной программы более чем на два порядка. Алгоритм умножения матриц на современных массовых процессорах описан в [**130**] и с разными уточнениями и реализациями входит в библиотеки BLAS.

Другие библиотеки, которые не входят в состав системного ПО, поставляемого с процессором, для достижения высокой эффективности необходимо переписывать. Наивные пользователи могут это не понимать и терять производительность.

Пакеты прикладных программ ориентированы на некоторые предметные области, содержат библиотеку решателей и оболочку (человеко-машинный интерфейс), позволяющую эти решатели настраивать на конкретные решаемые задачи. Пользоваться пакетами часто могут не являющиеся программистами специалисты в конкретных прикладных областях.

Пакет программ по линейной алгебре LAPACK основывается на библиотеке BLAS. Эта библиотека обычно входит в состав системного ПО. По этой причине данный пакет сохраняет высокую эффективность при переносе на дугой компьютер.

Кроме популярных пакетов, таких как **Wolfram Mathematica, MatLab, Maple, MathCAD, AutoCAD, ANSYS, ABAQUS, FlowVision** и **Gaussian** есть множество менее известных специализированных пакетов. Несколько десятков типовых вычислительных задач используются при моделировании прогноза погоды[14]. Решатели описанных пакетов теряют производительность при переносе на другие вычислительные архитектуры. Для такой переносимости решатели приходится переписывать. Эта проблема могла бы решаться хорошим оптимизирующим компилятором.

2.4. Методы оптимизации и распараллеливания

Параллельное программирование представляло собой сложный процесс. В [**131**] есть история создания параллельной программы,

[14]Модели прогноза погоды. https://windy.app/blog/what-is-a-weather-forecast-model-guide-on-forecast-models-all-around-the-world.html

начиная с момента, когда эта программа была написана, но не работала, к очевидно неправильно работающей версии, затем к версии, работающей с ошибками. На сегодняшний день появились инструменты разработки параллельных программ, которые существенно облегчают этот процесс.

Параллелизм в компьютерах используется исключительно ради одной цели — повышения быстродействия. Многообразие методов параллельного программирования диктуется многообразием параллельных архитектур суперкомпьютеров.

В.А. Вальковский предлагал методы распараллеливания программ на архитектуры с распределенной памятью [**132**, **133**] с минимизацией межпроцессорных пересылок. В этих работах распараллеливались тесные гнезда циклов с несложными истинными зависимостями по данным.

В [**134**] для минимизации межпроцессорных пересылок в распределенной памяти предлагаются такие приемы, как «дробление» (partitioning) и «выравнивание» (alignment).

В [**135**] представлен проект TAM (Threaded Abstract Machine) и методы разбиения программ на параллельно выполняемые нити (треды).

В [**136**] представлена параллельная машина Blue Gene и система параллельного программирования на нее Charm++, которая позволяет описывать параллельно выполняемые процессы (чары) и компилировать это описание в C++ с вызовами специальных функций, осуществляющих параллельное выполнение.

Необычные принципы программирования [**137**, **138**] предлагались для суперкомпьютера, основанного на ассоциативной памяти, который разрабатывался под руководством академика РАН В.С. Бурцева [**139**].

Для асинхронных вычислительных систем под руководством В.Е. Котова разрабатывался язык программирования на основе сетей Петри [**140**, **141**].

В [**142**] описан популярный в свое время язык OKKAM для транспьютерных вычислительных систем. Но новое поколение программистов для кластеров выбрало MPI [**143**–**145**].

Распараллеливание является частным случаем оптимизации (машинно-ориентированной) программ. С другой стороны, приемы оптимизации программ [**146**, **147**] для однопроцессорных компьютеров должны использоваться и при распараллеливании.

Первые существенные работы по автоматическому распараллеливанию были написаны в начале семидесятых годов. Здесь особенно следует отметить статьи Л. Лампорта [148, 149]. После этого по автоматическому распараллеливанию написано множество теоретических статей, книг, диссертаций и разработано много распараллеливающих программных систем, представленных, в частности, в обзорах [56, 66, 150–155].

Доступные студентам методы разработки параллельных программ создаются в ННГУ, например, [127].

2.5. Инструменты для разработки нейронных сетей

В сети[15] представлены популярные инструменты с открытым кодом для разработки нейронных сетей. В [156] рассмотрены рекуррентные нейронные сети для распознавания речи. В статье [157] предлагается фреймворк Acorns для ускорения глубоких нейронных сетей с разреженностью входных данных. В Acorns разреженные входные данные организованы в авторский разработанный макет разреженных данных, который обеспечивает удобный доступ к памяти для ядер в нейронных сетях и повышает производительность. В реальной задаче обнаружения при автономном вождении Acorns демонстрирует ускорение 1.8-2.4× по сравнению с Intel MKL-DNN [16], 3.0-8.8× с TensorFlow, и 11.1-13.2× с Intel MKL-Sparse[17]. В статье [158] исследуется разреженность нейронных сетей, возникающая из-за высокого уровня избыточности параметров сети. Предлагается инструмент ТензорКвант для исследования такой разреженности. Тензорквант может помочь в исследовании разреженности в глубоких нейронных сетях, определяя, где разреженность возникает в высокой степени. Информация, полученная из этого, может служить руководством для проектирования аппаратных ускорителей разреженной арифметики. TensorQuant является открытым исходным кодом и свободно доступен на GitHub5.

---

[15] Tensorflow фреймворк для нейросетей от Google с открытым исходным кодом URL https://www.tensorflow.org/xla, Pytorch фреймворк для нейросетей от Facebook с открытым исходным кодом URL https://pytorch.org/docs/stable/jit.html

[16] Библиотека для глубоких нейронных сетей Intel MKL-DNN URL https://software.intel.com/content/www/us/en/develop/articles/intel-mkl-dnn-part-1-library-overview-and-installation.html

[17] Developer Reference for Intel one API Math Kernel Library URL https://software.intel.com/content/www/us/en/develop/documentation/mkl-developer-reference-c/top/blas-and-sparse-blasroutines.html

В [159] рассматривается использование нейронных сетей для оптимизации в компиляторе. Предлагается встроить в компилятор трассировщик, собирающий информацию о прохождении трасс программой, на основе чего обучается нейронная сеть и оптимизируется выполнение. Данный подход предлагается использовать для программ с многими ветвлениями. Представлена реализация на основе LLVM.

2.6. Реконфигурируемые архитектуры и компиляторы для них

Много вычислительных устройств разрабатывается на ПЛИС — программируемых логических интегральных схемах [160–162]. Многоконвейерные модели вычислений рассматривались, например, в [99, 163, 164].

Всякая параллельная вычислительная архитектура не является универсальной. Программируемые (или перестраиваемые) архитектуры обладают возможностью относительных изменений. Часто перенастраиваются только связи между устройствами вычислительной системы. Идея программируемых (перестраиваемых) архитектур состоит в том, что не алгоритм решения задачи адаптируется к архитектуре, а, наоборот, вычислительная архитектура адаптируется к алгоритму решения задачи. Модели вычислительных систем с программируемой архитектурой обсуждаются несколько десятилетий.

Здесь можно отметить отечественные работы Э.В. Евреинова, В.Г. Хорошевского, А.В. Каляева, В.В. Корнеева: [27, 30, 31, 165–173]. Наиболее успешными оказались основанные на ПЛИС системы с архитектурой перестраиваемого конвейера [18, 166, 167, 174–179][18].

Описание функций, выполняемых проектируемой схемой, на языке программирования высокого уровня (Си, Паскаль, Фортран) требует меньше времени на разработку и отладку, чем на языках описания электронных схем (VHDL, VERILOG) [160, 162]. Представляется, что конвертор с языка высокого уровня C в HDL-язык сможет генерировать не только схемы цифровых фильтров и компрессоров/декомпрессоров [161], но и схемы конечных автоматов и программируемые вычислительные системы на ПЛИС (FPGA) [165, 166, 169, 171, 172, 174, 175].

[18] см. также Zynq-7000 All Programmable SoC Overview. Preliminary Product Specification. XILINX URL http://www.xilinx.com/support/documentation/data_sheets/ds190Zynq-7000-Overview.pdf.

Созданию компилятора на архитектуры программируемого конвейера посвящены работы [174, 175, 180, 181][19].

В [182–186] рассматривается проект компилятора с языка программирования высокого уровня Си на процессор с программируемым ускорителем. При использовании ПЛИС в качестве ускорителя много времени может отнимать перепрограммирование ПЛИС. Минимизация количества перепрограммирований рассматривалась в [187]. В работах [188, 189] рассматривается задача построения такого конвейера, который может выполнять функции нескольких конвейеров. В статье [190] получена классификация программных циклов, адаптированная к возможности конвейеризации (конвейерные вычисления наиболее успешно применяются на реконфигурируемых архитектурах).

Вероятно, в компиляторе на ПЛИС могут использоваться идеи совмещения программной конвейеризации внутреннего цикла с разверткой внешних циклов при компиляции гнезда вложенных циклов [191].

Вычисления на фотонном компьютере [54] подобны конвейерным вычислениям. Поэтому, разработка компилятора на фотонный компьютер может позаимствовать элементы компилятора на ПЛИС-ускоритель.

В работе [192] представлен язык программирования ПЛИС автокод HDL, имеющий более высокий уровень, чем VHDL и Verilog.

Язык COLAMO[20] для программируемых (на основе ПЛИС) вычислительных систем со структурно-процедурной организацией вычислений позиционируется как язык программирования высокого уровня с неявным параллелизмом. Основная единица языка — кадр, программа на языке COLAMO представляет собой последовательность кадров. Кадр определяет вычислительную структуру и потоки данных в реконфигурируемой вычислительной системе в конкретный момент времени. Все операции в теле кадра выполняются асинхронно с максимальным параллелизмом, а последовательность смены кадров определяется программистом. Пользователь языка COLAMO в кадре должен описывать структуру связей элементарных процессоров и ячеек памяти, в которых хранятся данные. Видимо, кадр соответствует одной схеме, которая прожигается на ПЛИС. Вероятно, перед каждым

[19] Компилятор C2H с языка Си на ПЛИС (Альтера) https://www.altera.com/en_US/pdfs/literature/ug/ug_nios2_c2h_compiler.pdf предполагал использование многих прагм, что требует от пользователя понимание многих моментов, которые язык высокого уровня должен скрывать. Этот проект уже не поддерживается.

[20] http://superevm.ru/index.php?page=yavucolamo

кадром ПЛИС заново прожигается. Описание данных сопровождается способами хранения (это не свойственно для языков высокого уровня), как во внешней (относительно ПЛИС), так и во внутренней памяти.

Обзор особенностей и перспектив развития программируемых и реконфигурируемых суперкомпьютеров на ПЛИС представлен в [193].

Следует подчеркнуть, что было бы сложно генерировать VHDL-код конвейерной системы из низкоуровневого регистрового внутреннего представления, какими являются LLVM компилятора Clang или RTL семейства компиляторов GCC. Высокоуровневым внутренним представлением (в котором производятся оптимизирующие преобразования) кроме OPC [194–199] обладает распараллеливающая система SUIF и семейство компиляторов ROSE.

Компилятор на реконфигурируемую архитектуру в самом общем случае включает в себя генератор вычислительных систем. В 90-ые годы наиболее эффективными были конвейерные вычислительные системы и в начале нулевых появились успешные вычислительные системы с архитектурой перестраиваемого конвейера. По мере того, как возрастала плотность микросхем (и, следовательно, быстродействие вычислительных операций) с большим опережением по отношению к производительности памяти начали обсуждаться многоконвейерные (мультиконвейерные) системы. С ростом количества вычислительных элементов на ПЛИС растет и сложность коммутации этих элементов. Ограниченное количество слоев микросхемы ПЛИС ограничивает возможности коммутации. Это приводит к потребности реализации на ПЛИС систолических структур (схема которых — удобный для аппаратной реализации плоский граф) [?59].

2.7. Выводы к главе

Для достижения высокой производительности во многих случаях оптимизирующие преобразования программ применяются вручную, требуя высокой квалификации программиста и больших затрат времени. Многие преобразования, которые делаются вручную, можно реализовать автоматически.

Для компьютеров с распределенной памятью либо предлагаются методы ручного распараллеливания, либо системы с частично автоматическим распараллеливанием. Нет методов автоматического распараллеливания на суперкомпьютеры с общей распределенной, памятью, такой, как у суперкомпьютера RP-3 [41] или суперкомпьютеров со структурно-процедурной организацией вычислений [166].

С другой стороны, в [200, 201] производительность автоматической оптимизации классических программ перемножения матриц и умножения матрицы на вектор приближается к библиотечной.

Все это говорит о необходимости и возможности развития автоматической оптимизации программ.

3. Классические преобразования программ

Об оптимизирующих преобразованиях написано много книг и обзорных статей [15, 87, 146, 147, 150, 153, 202–209]. Опытные компиляторщики говорят о существовании нескольких сотен преобразований. Можно преобразования разбить на группы в зависимости от того, какие части программ оптимизируются:

(1) Оптимизация выражений, линейных участков и базовых блоков.

(2) Оптимизация циклов.

(3) Оптимизация данных.

(4) Оптимизация функций.

(5) Оптимизация тесных гнезд циклов.

3.1. SSA-form для оптимизации выражений, линейных участков и базовых блоков

Оптимизацию выражений, линейных участков и базовых блоков удобно выполнять в SSA-form (Single Static Assignment form) — специальном внутреннем представлении фрагмента программы, удовлетворяющее принципу однократного присваивания, для оптимизации. В таком представлении упрощается необходимый анализ индуктивных переменных и анализ условий для выполнения подстановки, переименования, удаление мертвого кода, выноса общих подвыражений, понижение сложности операций, упрощения логических операторов и др. SSA-форма разработана в [210] и развивалась во многих работах, например, в [211, 212]. Используется в популярных оптимизирующих компиляторах с регистровым внутренним представлением GCC, LLVM. Эта SSA форма встречается в исследованиях по символьному анализу [213].

3.2. Примеры преобразования циклов

Циклы — основной объект оптимизации программ, поскольку, во многих случаях, циклы требуют больших объемов вычислений и

нуждаются в ускорении. Преобразования рассматриваются, в основном, для циклов со счетчиком: циклы DO языка ФОРТРАН и циклы FOR языков Pascal и C/C++.

Приведем краткие описания основных преобразований циклов (точные определения преобразований должны содержать описания условий выполнения).

- Разбиение цикла. Заменяет цикл, в теле которого много операторов, несколькими циклами, объединение тел которых составляет тело исходного цикла.
- Слияние циклов. Преобразование, обратное к разбиению. Два последовательно записанных цикла (с одинаковыми заголовками) заменяет одним циклом, тело которого является объединением тел исходных циклов.
- Расщепление цикла. Заменяет исходный цикл двумя, тела которых такие же, как у исходного, а пространства итераций являются разбиением пространства итераций исходного цикла.
- Раскрутка цикла. Цикл заменяется последовательностью блоков, каждый из которых является копией тела исходного цикла, а количество этих блоков равно количеству итераций исходного цикла.
- Развертка цикла кратности k. Цикл заменяется новым циклом, тело которого состоит из k блоков, каждый из которых является копией тела исходного цикла, а количество этих итераций результирующего цикла равно количеству итераций исходного цикла, деленному на k.
- Вынос инвариантов из цикла.
- Гнездование цикла с параметром M. Цикл заменяется гнездом из двух вложенных циклов, тело которого состоит из k блоков, каждый из которых является копией тела исходного цикла, а количество этих итераций результирующего цикла равно количеству итераций исходного цикла, деленному на k.
- Векторизация цикла. Заменяет последовательный цикл векторной командой (все итерации выполняются одновременно синхронно).
- Распараллеливание цикла. Заменяет последовательный цикл параллельным (все итерации выполняются одновременно асинхронно).

Распараллеливанию циклов посвящено много публикаций, например, [**214**–**222**]. Некоторые циклы непосредственно не распараллеливаются,

но могут стать распараллеливаемыми после специальных вспомогательных преобразований. Такими являются, например, циклы с линейной рекуррентной зависимостью [87, 217, 218].

Одно из наиболее сложных преобразований программ «разбиение цикла» (loop distribution) заменяет цикл, в теле которого много операторов, несколькими циклами, в теле каждого из которых подмножества операторов исходного цикла. Это преобразование описывается, например, в работах [15, 219, 223]. Условия преобразования цикла к разбиваемому виду близки к условиям преобразования цикла к векторизуемому виду. Здесь возникают задача построения фактор-графа графа информационных связей по компонентам сильной связности, NP-трудная задача [2] поиска наименьшего множества дуг, удаление которых делает граф бесконтурным. Классификация циклов с одним оператором присваивания представлена в [190, 216].

Перспективными выглядят работы, позволяющие сгруппировать в теле цикла операторы из разных итераций [220, 221, 224, 225] (при этом, за пределы цикла выносится несколько операторов).

3.3. Преобразования данных

В качестве вспомогательных преобразований для разбиения цикла или для векторизации цикла могут использоваться растягивание скаляров, повышение размерности массивов, введение временных массивов (см., например, [207, 219]). Более общим по отношению к растягиванию скаляров и повышению размерности массивов является экспансия массивов «array expansion» [226].

Выравнивание (alignment) оптимизирует кэш-линейки для считывания данных (см., например, [147]). Современные компиляторы могут делать выравнивание автоматически.

Преобразования структур для ускорения работы СУБД [227].

3.4. Преобразования функций

Оператор вызова функции (процедуры) компилятором преобразуется в большое количество ассемблерных операций: присваивание формальным параметрам фактических, переход на код описания функции, переход после вычисления функции на место ее вызова. Оптимизирующие компиляторы выполняют преобразование «инлайнинг», которое подставляет вместо вызова функции тело этой функции с соответствующими значениями параметров. Эффективность

инлайнинга рассматривалась, например, в [228–230]. Следует отметить, что применение инлайнинга к часто вызываемым вызовам функции может привести к увеличению программного кода, что, с некоторого момента, может вызвать замедление.

Иногда в программах встречаются вызовы функций с константными значениями некоторых параметров. В этом случае можно создать новую функцию, которая выполняет то же, что и исходная, но не имеет тех параметров, которые в вызовах имеют константные значения. При компиляции вызов новой функции создает меньше ассемблерного кода. Кроме того, константные значения параметров исходной функции могут позволить упростить некоторые вычисления на этапе компиляции. Эти преобразования функций относятся к таким направлениям теории программирования, как частичные, смешанные вычисления, суперкомпиляция см., например [231].

3.5. Композиции преобразований

Композиция преобразований программ является преобразованием. Тождественное преобразование, очевидно, является нейтральным элементом относительно композиции. У некоторых преобразований есть обратные. Например: разбиение цикла и слияние циклов; подстановка и вынос вычисления общих подвыражений.

Но преобразования не применимы ко множеству всех программ. У каждого преобразования есть свое множество программ, к которому оно может быть применимо. Преобразования применяются к программам при выполнении некоторых условий.

Бывают сложные оптимизирующие преобразования, которые представляют собой композицию нескольких более простых. При этом, входящие в композицию преобразования могут код замедлять, в то время как композиция код ускоряет. Например, перед применением векторизации к циклу могут применяться вспомогательные преобразования «введение временных массивов», «растягивание скаляров» и др. [207].

3.6. Выводы к главе

У одной программы могут быть конфликтующие пары преобразований: каждое из преобразований может быть применено к программе: либо одно, либо другое — но не оба. Например, выражение a*b+a*c+b*c можно преобразовать к виду a*(c+b)+bc либо к a*b+(a+b)*c, но

оба преобразования одновременно невозможны. Возникает задача выбора оптимального множества (последовательности) преобразований, применимых к данному коду [**232**]. Решение этой задачи может привести к перебору некоторого (большого) дерева вариантов. Такой перебор можно было бы и распараллелить, но это пока не делается.

Преобразование «ретайминг» одномерных (самых вложенных) циклов разработано для векторизации и получило обобщение для оптимизации используемых регистров. Наверное, для не самых вложенных циклов это преобразование можно было бы использовать для оптимизации использования кэш памяти.

4. Проверки применимости преобразований

4.1. Разновидности проверок

Перед применением преобразований выполняется ряд проверок.

(*1*) Проверка эквивалентности. Правда ли, что фрагмент, который предполагается оптимизировать, не содержит недопустимых информационных зависимостей?

(*2*) Проверка эффективности (целесообразности). Правда ли, что преобразованный фрагмент будет лучше исходного?

(*3*) Другие проверки. Проверка корректности. Правда ли, что фрагмент, который предполагается оптимизировать, не содержит недопустимых для преобразования единиц языка?

(*4*) Анализаторы для преобразований.

4.2. Проверки эквивалентности

Упрощенно говоря, две программы считаются эквивалентными, если на одних и тех же входных данных эти программы выдают одинаковые результаты. Более строго для эквивалентных программ следует определить, какие выходные результаты можно считать одинаковыми (например, насколько допустимо отличие вещественных чисел, какие перестановки слов допускаются в текстах, какие допустимы трансформации изображений и т.д.) и можно потребовать, чтобы у эквивалентных программ совпадали области определения.

Преобразование называется эквивалентным, если исходная и преобразованная программы эквивалентны.

Для проверки эквивалентности анализируется сохранение порядка обращений к памяти при выполнении преобразования. То есть, при

преобразовании программы порядок выполнения операций можно менять, если при этом для любых двух вхождений переменных, которые обращаются к одной и той же ячейке памяти (оперативной, регистровой или иной), так, что, хотя бы одно из вхождений является записью, не изменяется порядок обращений к памяти.

В [**233**–**235**] приводятся результаты статистических исследований, демонстрирующие высокий процент программ, в которых индексные выражения переменных аффинно зависят от счетчиков циклов. Информационные зависимости анализируются, как правило, для таких вхождений переменных.

Информационные зависимости между вхождениями переменных или операторами делятся на истинные (или потоковые), антизависимости, выходные и входные, Если вхождения или операторы оказываются внутри цикла, то зависимости могут быть циклически независимыми (loop independent) или циклически порожденными (loop carried). [**15**, **87**, **199**, **205**, **206**].

Еще иногда используется понятие ложной зависимости. Ложная зависимость между вхождениями (v1,v2) — это истинная зависимость (такое парадоксальное определение!), которая не реализуется, то есть вхождение v2, читающее данную ячейку памяти, никогда не читает значение, записанное в эту ячейку вхождением v1 (а читает значение, записанное другим вхождением переменной).

Контроль за сохранением порядка обращений к памяти ведется, как правило, с помощью графовых моделей информационных зависимостей. К таким моделям относятся граф информационных связей (вершины графа — вхождения переменных), граф зависимостей по данным (вершины графа — операторы программы), решетчатый граф программы (вершины графа — точки пространства итераций гнезда циклов). В работах P. Feautrier и L. Lamport слово «граф» не используется, но в неявном виде присутствует. В.В. Воеводин, использовавший в первых работах термин «решетчатый граф», впоследствии заменил его термином «граф алгоритма».

Диссертация [**236**] содержит большой обзор публикаций по анализу информационных зависимостей (неравенства Банерджи, Омега-тест [**237**] и пр.).

Точная проверка эквивалентности может упираться в анализ псевдонимов (некоторый обзор алгоритмов представлен в [**238**] в главе 1) и требовать чрезмерно большого времени. Как правило, для анализа псевдонимов используются эвристические алгоритмы.

4.3. Проверка целесообразности (эффективности) преобразования

Такие проверки должны предугадывать, станет ли после преобразования программа лучше. Для многих преобразований такая проверка очень сложна. Часто в компиляторах пользователю предлагается написать прагму с указанием на выполнение преобразований (например, распараллеливание). Иногда компиляторы на этапе компиляции могут выделить исходный и преобразованный фрагменты кода в самостоятельные программы, откомпилировать, измерить и сравнить время выполнения. Либо предполагается какая-то модель времени выполнения (какая операция или оператор как долго выполняются).

Ранее, в 60-ые годы прошлого века, время работы программы (алгоритма) оценивалось количеством вычислительных операций. В последние годы время работы программы оценивается количеством обращений к памяти. Статический анализатор [**238**–**240**] оценивает у фрагмента программы не только количество вычислительных операций, но и объем используемых данных. Этот анализатор, видимо, может быть развит для прогноза эффективности преобразований.

Производительность компиляторов и процессоров оценивается тестированием на бенчмарках: Spec[21], NAS[22], PolyBench[23] и один из первых наборов тестирования производительности — Ливерморские циклы[24].

4.4. Диалоговое уточнение зависимостей

В работе [**213**] показано, что алгоритм Флойда не может быть автоматически распараллелен, и предлагается использовать диалог с пользователем для распараллеливания этого алгоритма в случае неотрицательной входной матрицы весов графа (компилятор языка C может знать только тип данных, но не их диапазоны, которые влияют на информационные зависимости). Диалог формируется на основе символьного анализа текста программы.

---

[21]Standard Performance Evaluation Corporation URL https://spec.org/.

[22]NAS Parallel Benchmarks URL https://en.wikipedia.org/wiki/NAS_Parallel_Benchmarks.

[23]L.-N. Pouchet PolyBench/C the Polyhedral Benchmark suite URL http://web.cse.ohio-state.edu/~pouchet.2/software/polybench/.

[24]Livermore loops URL https://en.wikipedia.org/wiki/Livermore_loops.

Диалоговая распараллеливающая компиляция — это создание параллельной программы на основе последовательной в диалоговом режиме. Это полуавтоматическое распараллеливание (см., например, [241]). Интерактивное распараллеливание [242] пользователю позволяет в тексте программы писать команды компилятору, но не указывает, в каком месте и какую команду.

Преобразования программ в диалоговой системе [243] должны высокоуровневый текст преобразовывать в высокоуровневый текст, поскольку выходной текст должен быть узнаваемым пользователем. По этой причине для диалога удобнее проводить преобразования в высокоуровневом внутреннем представлении, из которого генерация высокоуровневого текста менее искажается, чем из низкоуровневого.

Обычная компиляция не использует ассоциативность операций сложения и умножения вещественных чисел, поскольку изменение порядка выполнения операций может привести к изменениям погрешностей или переполнениям. Но разработчик программы может знать, когда изменение порядка выполнения операций над вещественными числами допустимо и не приведет к нежелательным последствиям. Поэтому в диалоге система может ему позволить выполнение таких рискованных операций (м.б. с предупреждениями). В частности, благодаря изменению порядка выполнения операций можно распараллеливать (с санкции разработчика) рекуррентные циклы [100, 217, 218].

4.5. Анализ корректности преобразований

При неосторожном использовании преобразований в результирующей программе могут возникать нарушения синтаксиса и семантики языка. Например, рассмотрим раскрутку или развертку цикла. Предположим, что в исходной программе в теле цикла был оператор с меткой. В результате (неосторожно выполненной) раскрутки или развертки появляется несколько копий тела цикла. Не получится ли в разных копиях тела цикла несколько операторов с одинаковыми метками? Или после раскрутки цикла программы языка Си не получится ли оператор Continue или Break не внутри цикла? Как застраховаться от таких ошибок? В этом направлении можно отметить работу [244], метод которой, в отличие от тестирования, гарантирует отсутствие ошибок некоторого класса — начало исследований, не доведенное до завершения.

Система перебора комбинаций преобразований и уточнение информационных зависимостей представлены в [97]. Тестирование блока

преобразований представлено в [90, 92]. Известна диалоговая система для отладчиков [245].

4.6. Выводы к главе

Проверки применимости необходимы для автоматизации преобразований. Некоторые проверки выполнить автоматически на этапе компиляции теоретически невозможно, но можно было бы делать в диалоговом режиме уточнения зависимостей, которое пока в стадии исследований, у компиляторов его нет. Интерактивный режим (написание пользователем компилятора прагмы) слабее диалогового, поскольку требует от пользователя понимания, в каком месте программы и какую прагму следует писать. Анализ (в том числе, и символьный) количества обращений к памяти (каждого уровня) мог бы позволить прогнозировать время работы фрагмента кода и, этим самым, прогнозировать эффективность преобразования. Вычислительно сложный анализ псевдонимов можно было бы распараллелить. Отмеченные замечания демонстрируют потенциал развития компиляторов.

5. Тайлинг, локализации данных и распараллеливания тесных гнезд циклов

5.1. Тайлинг и локализация данных

Тайлинг [246] — это переход к блочным вычислениям в тесных гнездах циклов. Если имеется тесное гнездо длинных циклов, то необходимые для вычисления этого гнезда данные могут не помещаться в кэш-память, что вызывает обмены данными между кэш и оперативной памятью. Тайлинг преобразует это гнездо к новому гнезду с большим количеством циклов, у которого несколько самых внутренних циклов короткие. Такое преобразование уменьшает обмены данными между модулями памяти разного уровня: между L3-кэш и оперативной памятью, L2-кэш и L1-кэш. Более того, если у производительности программы узким местом был доступ к данным оперативной памяти, то тайлинг приводит программу к виду, в котором с данными, попавшими в кэш-память, можно проводить много вычислений и делает целесообразным различные ускорения этих вычислений, в частности, распараллеливание. Аналогично, тайлинг может уменьшить количество промахов к L1-кэш, повысив, заодно, эффективность векторизации.

Локализация данных — это такой способ организации вычислений, при котором одни и те же данные, попав в кэш, многократно используются. Локализация данных ориентирована на уменьшение количества кэш-промахов. Локализация бывает временная и пространственная. Временная локализация предполагает близкое по времени использования размещение в исполнимом коде операций с одинаковыми данными (в этом случае, при повторном обращении к данному это данное может оказаться в кэш-памяти). Пространственная локализация предполагает близкое размещение в оперативной памяти данных, которые используются программой в близкое время (в этом случае, в кэш-линейке кроме необходимого в текущий момент данного, может содержать и данное, которое потребуется в близкое время).

Локализацию данных может улучшать слияние циклов, если сливаемые циклы содержат одинаковые переменные. Использование слияния циклов для локализации данных отмечено в работах [**247**, **248**]. Однако, в этих работах не рассмотрены случаи, при которых сливаемые циклы связаны информационными зависимостями относительно вышестоящих циклов. Также, не рассматривается использование цепочки вспомогательных преобразований (таких как “введение временных массивов”, “растягивание скаляров”, “перестановка операторов”) для слияния. Сложные примеры слияния представлены в [**249**].

В работе [**247**] рассматриваются как тайлинг, так и слияние циклов.

Актуальность оптимизации параллельных вычислений подчеркивается тем, что параллельные программы часто показывают низкую эффективность [**250**]. Для многих приложений наблюдается, что при повышении уровня параллелизма с некоторого момента скорость выполнения программ замедляется. Проблема отставания реальной производительности программного обеспечения от пиковой производительности вычислительных систем отмечается как отечественными, так и зарубежными исследователями [**57**, **251**, **252**]. Это во многих случаях объясняется тем, что вычислительная система не успевает готовить данные для вычислений. Локализация данных может решать эту проблему.

Применение принципа построения блочных алгоритмов для обработки сигналов и изображений для графической карты представлено в [**253**].

В [**227**] представлено преобразование «расщепление структур» (structure splitting). В используемых в программе структурах выделяются поля, обращения к которым встречаются вместе. Такие

поля объединяются в одну структуру, обращение к которой повышает локальность данных. На двух тестах из пакетов SPEC CPU2000 и SPEC CPU2006 получено ускорение на 19% и 12%.

5.2. Двойной тайлинг и блочные размещения массивов

Поскольку иерархия памяти у многих процессоров имеет много уровней, возникает естественная идея многоуровневого тайлинга [**254**]. Но несложно заметить, что переход к тайлингу приводит к более сложным индексным переменным массивов. Усложнение вычисления адресов могут вызывать замедление. Два уровня тайлинга дают ускорение, но третий уровень оказывается неэффективным. Двойной тайлинг демонстрирует эффективность [**130**, **200**, **255**].

Матрица в программе описывается как двумерный массив, а оперативная память для программиста представляется одномерной линейной последовательностью адресов. По стандартам языка ФОРТРАН матрицы размещаются в одномерной памяти по столбцам, а по стандартам языка Си — по строкам. Эти стандарты существуют давно, до появления кэш-памяти и кэш-линеек в процессорах.

Если матрицу разбить на блоки и размещать в одномерной оперативной памяти так, что элементы каждого блока будут находиться рядом, то количество промахов к кэш-памяти (включая дорогие промахи к TLB-кэш) может уменьшиться. Блочные и двойные блочные размещения массивов рассматриваются в работах [**214**, **255**, **257**–**259**].

Следует отметить, что нестандартные блочные размещения массивов затрудняют их использование в библиотеках. Для использования такой библиотечной программы возникает необходимость в переразмещении данных, что может отнимать время.

5.3. Модели информационных зависимостей в гнездах циклов. Решетчатые графы

Вершины решетчатого графа — это точки пространства итераций гнезда циклов. Дуга соединяет две таких вершины, если в копиях тела гнезда циклов, соответствующих этим точкам, происходит обращение к общим ячейкам памяти. То есть, если сделать раскрутку всех циклов гнезда, то между копиями тела будет дуга графа информационных связей. Поскольку дуги бывают разных типов (истинной, анти-, выходной, входной) зависимости, то и решетчатые графы могут быть таких же типов. Изначально рассматривались только дуги

истинной зависимости, из множества которых удалялись дуги ложной зависимости.

Решетчатые графы использовались в методах распараллеливания программ с семидесятых годов в работах Л. Лампорта в методе гиперплоскостей [**149**], В. Вальковского в методе параллелепипедов [**133**] и методе пирамид [**132**] и у других исследователей. Эти графы использовались для иллюстрации идей, лежащих в основе методов распараллеливания. В явном виде эти графы не использовались, поскольку традиционные формы хранения графа в виде матрицы смежностей или в виде списка дуг не эффективны: прочтение такого графа может потребовать больше времени, чем последовательное исполнение программы, по которой этот граф построен. В конце 80-х годов был сделан прорыв в этом направлении: два математика В.В. Воеводин [**36**, **223**, **260**, **261**] и P. Feautrier [**226**, **262**–**264**] научились хранить этот граф в компактном виде, в виде функций, и разработали алгоритмы построения этих функций. У P. Feautrier функция содержит в своем определении не очень удобные для исследования условные операторы. У В.В. Воеводина функция является кусочно-линейной. В теории решетчатых графов можно отметить также работы Н.А. Лиходеда [**265**], А.В. Фролова [**71**, **165**] и работы в группе ОРС (проект Открытая (Оптимизирующая) распараллеливающая система) [**84**, **91**, **93**, **95**–**98**, **266**].

В работе [**266**], которая применима к узкому классу программ, на основе анализа зависимостей между точками пространства итераций оценивается минимальное количество шагов параллельного выполнения гнезда циклов.

В работе [**267**] показана связь между решетчатым графом и графом информационных связей программы, полученной из гнезда циклов полной раскруткой всех его циклов.

Анализ зависимостей между точками пространства итераций в гнезде циклов позволяет распараллеливать вычисления точек пространства итераций даже в тех случаях, когда ни один цикл гнезда не может быть распараллелен. Неунимодулярные преобразования пространства итераций, которые могут привести к появлению распараллеливаемого цикла, представлены в [**268**]. Известно, что параллельное выполнение точек гиперплоскостей пространства итераций может приводить к замедлению, но, если перейти к блочному коду (тайлинг), и параллельно выполнять блоки точек пространства итераций, то ускорение от распараллеливания будет приближаться к оптимальному [**215**].

5.4. Распараллеливание гнезд циклов

Распараллеливающая система Pluto [269–271] выполняет линейные (аффинные) преобразования пространства итераций гнезда циклов. В [272] рассматривается аналог Pluto и сравнение этих систем. Но система Pluto модифицируется [273]. Для анализа информационных зависимостей между точками пространства итераций используется теория целочисленного линейного программирования P. Feautrier и основанная на ней библиотека PipLib. На основе анализа зависимостей в пространстве итераций выбирается гиперплоскость (волновой фронт), точки которой могут выполняться одновременно. Линейные преобразования позволяют изменить обход точек пространства итераций так, что этот обход описывается новым гнездом циклов, у которого один или несколько циклов сканируют целевую гиперплоскость и допускают эффективное распараллеливание или векторизацию. Линейные преобразования могут быть неунимодулярными, в частности, описывается преобразование скашивания (skewing)

В LLVM есть фреймворк Polly, который использует такую же полиэдральную модель, как и Pluto. Компиляторами семейства LLVM можно преобразовывать гнезда циклов с помощью Polly, но для этого компиляторы должны получать много указаний, поскольку Polly не определяет оптимальные размеры тайлов и пр.

Тайлинг можно свести к преобразованию «перестановка заголовков циклов в тесном гнезде». Условия корректности такого преобразования описаны в статье Вольфа и Банерджи [274] в терминах специальной алгебры отношений. Следует отметить, что этот подход позволяет рассматривать тайлинг только прямоугольный (разбиение пространства итераций на прямоугольные параллелепипеды).

Наилучшего быстродействия достигают программы, которые одновременно являются параллельными и учитывают иерархию памяти [130, 214, 215, 257, 275–281]. В частности, в работах [281–286] рассматривается непрямоугольный тайлинг (ромбический), который (сделанный вручную) в сочетании с распараллеливанием, дает ускорения в несколько раз (иногда более, чем на порядок!) для сеточных методов решения различных классических уравнений математической физики.

Автоматическое разбиение пространства итераций на блоки и параллельного выполнения таких блоков делается пока только для простейших случаев (например, для задач перемножения матриц и умножения матрицы на вектор [200, 201]). Для нетесных гнезд циклов мелкозернистое распараллеливание представлено в [287].

5.5. Узкие места производительности

В течение нескольких десятилетий наблюдается рост производительности процессоров (выполнение операций умножения) примерно на 30% в год, а рост пропускной способности памяти (скорость чтения/записи данных из оперативной памяти) растет только на 9% [**288**]. Таким образом, если ранее основное время работы вычислительных систем уходило на вычислительные операции, то сейчас — на доступ к данным. Сложность алгоритма по обращениям к данным рассмотрена, например, в [**289**].

Наиболее типичная последовательность прохождения данных в компьютере с общей памятью, иерархией кэш-памяти и многоядерным процессором выглядит примерно следующим образом:

RAM -> L3 -> L2 -> L1 -> R -> ALU

Распараллеливание (OpenMP) происходит после L3 — у каждого процессорного ядра свой L2-кэш. Векторизация происходит после L1 через векторные регистры.

Для компьютеров с распределенной памятью добавляется коммуникационная сеть или коммутатор

NET -> RAM -> L3 -> L2 -> L1 -> R -> ALU

Каждая стрелка означает передачу данных: чем стрелка левее, тем передача дольше.

У разных программ узкое место производительности может быть в разных местах. При этом используются разные приемы оптимизации. В частности, если в программе самые глубоко вложенные циклы имеют в теле много операций с небольшим объемом данных, то вполне интересна оптимизация регистров. Но большинство используемых современных программ имеют много данных, с которыми выполняется относительно немного операций. Для таких программ узким местом является доступ к памяти.

Тайлинг — одно из наиболее используемых и эффективных преобразований, уменьшающих количество кэш-промахов.

5.6. Выводы к главе

Тайлинг делается для тесных гнезд циклов. Не исследованы автоматические преобразования нетесных гнезд циклов к тесным.

Для гнезд циклов с относительно сложными зависимостями (но с константным дистенс-вектором) тайлинг (ромбический) делается

вручную с применением высокого «искусства программирования ЭВМ» и приводит к повышению производительности более, чем на порядок (например, в работах В. Левченко). Автоматически это не делается.

6. Вычислительные системы с распределенной памятью и компиляторы для них

До появления многоядерных процессоров было много попыток разработки автоматического распараллеливания на вычислительные системы с распределенной памятью. Обзор 18 таких (давних) экспериментальных отечественных систем представлен в [**56**].

Для многих задач обмены данными стали узким местом при распараллеливании в системах с распределенной памятью [**290**]. В работах [**258**, **259**] описаны модели вычислений простых алгоритмов для архитектур с распределенной памятью. Рассмотрена простая задача, для выполнения которой на системе с распределенной памятью увеличение количества процессорных элементов сначала ведет к ускорению, а затем, к замедлению. Оптимальное количество процессорных элементов зависит от соотношения времени пересылки данных ко времени выполнения вычислительной операции.

Для вычислений на системах с распределенной памятью, как правило, минимизируют время межпроцессорных пересылок [**291**]. Это означает, что неявно время вычислений определяют, как количество пересылок данных. Размещения данных с перекрытием [**276**, **279**, **281**, **292**] уменьшают количество межпроцессорных пересылок данных.

Преобразование последовательных программ на вычислительные системы с распределенной памятью требует разработки дополнительных функций, которых нет в системах с общей памятью: размещение данных в распределенной памяти, переразмещения данных в распределенной памяти, генерация межпроцессорных пересылок, если вычислительная система имеет коммутатор, то автоматическая его настройка на выполнение необходимых пересылок.

Библиотека MPI предполагает, что каждый узел системы имеет все данные. Это создает следующие проблемы: 1) все данные в одном модуле памяти могут не помещаться; 2) если данные помещаются, то требуется время на их размещение во всех узлах; 3) в результате вычислений, как правило, могут возникать данные, представленные не в каждом узле — для дальнейших вычислений их следует переразмещать. Иногда удается обманывать библиотеку MPI, размещая в каждом

модуле памяти разные данные, под одним именем. Для автоматической генерации кода, наверное, лучше вместо MPI использовать SHMEM.

В работе [293] отмечается «Компиляция для параллельной архитектуры с распределенной памятью считается очень сложной задачей. ... никакого практического и эффективного решения в настоящее время не существует.». Генерируется код с генерацией коммуникаций, основанных на MPI (для кластера с мультипроцессорами). Компилируются гнезда циклов с аффинными зависимостями. Об оптимизации размещения данных не говорится ничего. Используется полиэдральная распараллеливающая система Pluto. Ничего не говорится о локализации данных. «Распределение данных и распределение вычислений тесно связаны — изменение распределения данных простым способом часто требует полной перезаписи вычислительного и коммуникационного кода.».

В [294] используется избыточное хранение некоторых областей данных. Размещение с оптимизацией пересылок. Неясно, как далеко может простираться архитектура разделяемой памяти, и на чипе многоядерные машины развиваются в сторону архитектуры распределенной памяти. Поэтому разработка методов архитектуры распределенной памяти является важным следующим шагом. Рассматривается параметрический тайлинг. дополнительная память для буферизованной и агрегированной коммуникации необходима по соображениям производительности. Отмечена эффективность сочетания тайлинга с распараллеливанием. Иллюстрации на примере алгоритма Смитта-Уотермана (выравнивание последовательностей). Алгоритм характерен большим объемом вычислений при малом количестве исходных данных. Т.е., пересылки занимают относительно небольшое время. Уделяется внимание анализу информационных зависимостей при организации распараллеливания.

В [295] сравнение возможностей языков параллельного программирования для вычислительных систем с распределенной памятью.

В [205] содержится обзор методов автоматического распараллеливания, актуальный для того времени (2004 г.).

Следует отметить, что на вычислительные системы с распределенной памятью генерируют параллельный код распараллеливающие системы Parawise[25] и DVM-система [104–124], при этом, прогноз эффективности распараллеливания ложится на пользователя системы. Размещение данных в распределенной памяти либо также ложится на пользователя, либо предполагается, что каждый узел кластера

[25]ParaWise Parallelizing System http://www.parallelsp.com/.

содержит все данные (предположение библиотеки MPI). Эти же замечания относятся и к работам [**296**, **297**].

Способы размещения данных в распределенной памяти рассматриваются в [**298**, **299**]. Блочно-аффинные размещения данных в распределенной памяти рассматриваются в [**81**, **259**, **291**], а алгоритмы их переразмещения рассмотрены в [**80**, **82**] — эти алгоритмы и методы могут быть использованы для автоматизации распараллеливания на системы с распределенной памятью.

Высокопроизводительные вычислительные системы с распределенной памятью — это, на сегодняшний день, в основном, суперкомпьютеры. Суперкомпьютеры — это дорогие не массовые компьютеры. Затраты на разработку высокопроизводительного прикладного ПО для таких компьютеров не очень велики по сравнению с ценой суперкомпьютера. Это одна из причин, по которой для таких компьютеров пока не развиты системы автоматизации разработки высокопроизводительных программ.

Появляются процессоры (системы на кристалле), которые можно рассматривать как кластер на одной микросхеме [**48**, **49**]. В этих системах время на пересылки между модулями памяти вычислительных ядер не так сильно отличается от времени вычислительных операций, как на высокопроизводительных кластерах (поскольку пересылки выполняются на микросхеме, а не между микросхемами). Развитие этих систем сдерживается отсутствием распараллеливающих компиляторов.

6.1. Выводы к главе

Высокую производительность демонстрируют многоядерные системы на кристалле, ядра которых имеют свою адресуемую (не кэш) локальную память. Адресуемая память на кристалле процессора (например, GPU или ускорители Intel Xeon Phi) демонстрирует производительность более высокую, чем у традиционных процессоров. Это можно объяснить не только большим количеством ядер, но и уменьшением обращений к оперативной памяти при операциях записи (действительно, всякое данное, которое находится в кэш-памяти, обязательно должно попасть через шину в микросхему оперативной памяти, но это не должно происходить при записи в локальную адресуемую память). Распределенная память на кристалле позволяет уменьшить время обращения к памяти на кристалле. Хотя в литературе и предлагаются способы описания размещений данных в распределенной

памяти, пока нет алгоритмов, которые по тексту программы определяли бы оптимальные размещения и команды переразмещения.

Следует отметить, что для распределенной памяти кластеров задача автоматического оптимального размещения данных не так актуальна, как для систем на кристалле. Это связано с различным отношением времени пересылки данных ко времеии их обработки. Для кластера это отношение намного больше, чем для системы на кристалле. Это означает, что для получения эффективности от распараллеливания на тот же объем вычислительных операций, для которого на кластере можно позволить одну пересылку, для системы на кристалле можно их позволить несколько. Малое количество пересылок на кластере можно писать вручную и это эффективно для не очень широкого класса программ. При большом допустимом количестве пересылок на кристалле расширяется класс программ и становятся актуальными задачи автоматической оптимизации этих пересылок.

7. Перспективы развития автоматической оптимизации программ

7.1. Границы возможностей оптимизации и распараллеливания

Существуют разрозненные результаты, описывающие границы возможностей некоторых отдельных фрагментов программ на некоторых вычислительных архитектурах. Например, в работах [**266**,**300**] делаются оценки количества одновременно выполняемых арифметических операций в гнезде циклов, при условии неограниченности количества вычислительных устройств и игнорировании времени на подготовку (в том числе и пересылки) данных. В [**80**,**82**] получены точные формулы минимального количества пересылок данных для тесного гнезда циклов и линейных индексных выражений переменных для МВС с распределенной памятью и универсальном коммутатором. Там же есть формулы и для случая кольцевой коммуникационной сети, но минимальность не доказана.

Чем больше процессорных элементов (ПЭ) - тем, с одной стороны, меньше тактов с арифметическими операциями (одновременными), но, с другой стороны, больше пересылок данных. Это подтверждает эксперимент, проведенный в 1989 году на ЭВМ ПС-2000 [**3**]. Тот факт, что из-за пересылок данных распараллеливание может привести не к ускорению, а к замедлению в задаче вычисления суммы N чисел отмечался в [**301**]. Задача оптимального использования ресурсов при

параллельных вычислениях рассматривалась в [302]. Но в этой работе программы представляются в виде графовой модели [36]. В частности, рекуррентные циклы по такой методике должны выполняться в одном ПЭ. В работах [87, 303] получены формулы оптимального количества процессорных элементов с учетом пересылок данных на МВС.

Некоторые циклы с линейной рекуррентной зависимостью удается преобразовать к виду, при котором возможно распараллеливание [86, 100]. Но, если рекуррентный цикл содержит условный оператор, условное выражение которого рекуррентно перевычисляется, то такой цикл можно распараллеливать, по сути, только в тех случаях, когда этот цикл вычисляет минимальный или максимальный элемент массива [304].

Современные компиляторы изменяют алгоритм в очень узких пределах. Оптимизация программ сводится к распознаванию кода, который можно оптимизировать, и замене найденного неоптимального кода более оптимальным. Эффективность преобразования может зависеть от целевой ВС. Например, на одних компьютерах стандартное перемножение матриц выполняется быстрее алгоритма Штрассена [305, 306], [5, стр. 220], а на других — наоборот. Распознать, что фрагмент программы выполняет перемножение матриц, особенно, если это алгоритм Штрассена, очень сложно. Более того, вычислительная сложность задачи перемножения матриц пока не определена. Есть результаты, позволяющие сомневаться в том, что самый эффективный алгоритм перемножения матриц существует: в [307] указывается существование такой задачи, у которой для любого алгоритма решения существует более быстрый алгоритм. Поэтому, вопрос о наилучшем ускорении программ, вряд ли разрешим в общем случае.

На данном этапе развития информационных технологий моделью времени выполнения алгоритма может служить отношение количества операций к количеству данных. Например, если вычисление суммы чисел быстрее считывания аргументов этого вычисления, то распараллеливание суммы элементов массива на общей памяти бессмысленно.

Площадь на кристалле (множество элементов микросхемы) можно по-разному распределить между вычислительными элементами (ядрами) и локальной (кэш) памятью. Для одних алгоритмов выгоднее одно соотношение памяти и вычислительных возможностей, а для других задач — другое [308]. Для алгоритмов с разными отношениями

количества операций к количеству данных оптимальны по эффективности разные ВС. Таким образом, в характеристике алгоритма «отношение количества операций к количеству данных» скрыты границы возможностей его распараллеливания: на одной ВС узким местом производительности будут вычисления, а на другой — доступ к данным.

Ведутся исследования [309] о возможном переходе от вычислений с вещественными числами к вычислениям в конечных полях. Пока не описаны границы (условия) эффективности такого перехода.

7.2. Потенциал современных оптимизирующих преобразований

Рассмотрим подробнее потенциал современных оптимизирующих компиляторов

- Автоматическое блочное размещение массивов в оперативной памяти.
- Определение по тексту программы оптимальных размещений и переразмещений массивов в распределенной памяти.
- Использование статического анализатора объема данных с использованием символьных вычислений для прогноза времени выполнения кода и определения оптимальных параметров преобразований (тайлинга).
- Анализ и параллельный обход дерева вариантов возможных комбинаций преобразований для поиска лучшей оптимизации.
- Повышение локализации данных за счет нового поколения преобразований циклов, включающего разбиение, развертки, раскрутки, слияние и ретайминг для не самых вложенных циклов со сложными зависимостями.
- Развитие компиляторов на программируемые архитектуры.
- Диалоговое уточнение информационных зависимостей.
- Автоматический ромбический тайлинг с проверкой информационных зависимостей.
- Преобразования нетесных гнезд циклов к тесным для последующего тайлинга и распараллеливания.
- Оптимизация и распараллеливание фрагментов кода, описанных рекурсивными функциями.

7.3. Алгоритмы, которые пока автоматически не ускоряются.

7.4. Итерационные алгоритмы

Много итерационных алгоритмов используется для решения задач, требующих большого объема вычислений. При этом, для одной и той же задачи несколько разных алгоритмов могут давать одно и то же решение (с точностью до допустимой погрешности). Одни такие алгоритмы могут быть более быстрыми, а другие менее; одни алгоритмы могут допускать хорошее распараллеливание, а другие - нет. Различные итерации одного итерационного алгоритма не могут выполняться независимо. Но иногда можно одновременно выполнять некоторые операции одной итерации с некоторыми операциями другой итерации [**281**].

Компиляторы не распознают итерационные алгоритмы. Распознавая итерационные алгоритмы можно было бы заменять один алгоритм другим или применять оптимизации методами, подобными [**281**].

7.5. Алгоритмы обхода дерева

Распараллеливание алгоритмов, основанных на методе ветвей и границ не всегда достаточно эффективно [**310**–**312**]. Много публикаций по распараллеливанию метода ветвей и границ или других алгоритмов, связанных с обходом дерева вариантов. Идея этих распараллеливаний состоит в одновременном обходе разных ветвей дерева вариантов. Однако, хотя в этих алгоритмах, как правило, очень высокое отношение количества операций к количеству используемых входных данных, далеко не всегда такое распараллеливание достаточно эффективно [**312**]. В работах [**241**, **311**] предлагается использование очереди заданий для обхода дерева вариантов, но этот инструмент не всегда дает приемлемое ускорение и не применяется автоматически.

По автоматическому распараллеливанию алгоритмов обхода дерева публикации не встречаются.

7.6. Алгоритмы выравнивания последовательностей

Алгоритмы выравнивания последовательностей имеют высокую алгоритмическую сложность $O(n^2)$. В этих алгоритмах на одно входное данное приходится много операций $O(n)$, что является поводом для ускорения с помощью тайлинга, распараллеливания и векторизации. В основе быстрых алгоритмов выравнивания лежат классические

алгоритмы парного глобального и локального выравниваний Смитта-Уотермана и Нидлмана-Уолша, которые сами выполняются медленно, поскольку используют $O(n^2)$ промежуточных данных, которые одновременно в кэш-памяти поместиться не могут. Для преобразования таких программ (алгоритмов) к блочной форме *приходится немного обобщать задачу:* на входе (выходе) программных блоков нужно иметь не только части выравниваемых (выходных) последовательностей, но и цену частичного выравнивания, приводящего к точке на границе блока (промежуточных) данных. Сочетание такого подхода с параллельным (OpenMP) выполнением блоков представлено, например, в [**215**, **313**]. Использование SIMD-ускорителей для задач выравнивания представлено, например, в [**128**, **314**–**316**], а также сопроцессоры Xeon Phi [**185**]. В [**317**] представлена программа, комбинирующая вычисления на центральном процессоре, графических ускорителях и сопроцессорах Xeon Phi.

7.7. Заключение

В последней главе и в выводах к другим главам формулируются задачи, решение которых может привести к созданию компиляторов нового поколения. Выделим направления развития оптимизирующей компиляции:

- Оптимальные размещения массивов в распределенной памяти.
- Автоматический параллельный обход дерева вариантов.
- Автоматический ромбический тайлинг.
- Блочное размещение массивов в оперативной памяти.
- Символьные вычисления для прогноза времени выполнения кода.
- Локализация данных за счет новых преобразований циклов.
- Развитие компиляторов на программируемые архитектуры.
- Ускорение работы компиляторов
- Диалоговое уточнение информационных зависимостей.
- Оптимизация и распараллеливание итерационных алгоритмов
- Переход к обобщенным задачам при оптимизации.
- Разработка комбинаций преобразований

Преобразования многих программ, проводимые «вручную» демонстрируют потенциал их ускорения «в разы» и даже «на порядок». Таким образом, для ускорения разрабатываемой прикладной программы м.б. вместо (или вместе) покупки нового компьютера приобрести новый перспективный оптимизирующий компилятор?

Развитие оптимизирующей компиляции приведет к сокращению времени разработки эффективных программ, к стимулированию развития новых более сложных вычислительных систем и новых приложений.

Список литературы

[1] *Большие задачи и суперЭВМ*, Пер. с англ., Труды института инженеров по электротехнике и радиоэлектронике, т. **77**, №7, Мир, М., 1989. ↑23

[2] М. Гэри, Д. Джонсон. *Вычислительные машины и труднорешаемые задачи*, Мир, М., 1982, 416 с. ↑23,42

[3] Ю. Затуливетер, Е. Фищенко. «Многопроцессорный компьютер ПС-2000», *Открытые системы. СУБД*, 2007, №9, с. 74–79. URL ↑24,57

[4] H. J. Siegel. "A model of SIMD machines and a comparison of various interconnection networks", *IEEE Transactions on Computers*, **C-28**:12 (1979), pp. 907–917. doi ↑24

[5] И. В. Прангишвили, С. Я. Виленкин, И. Л. Медведев. *Параллельные вычислительные системы с общим управлением*, Энергоатомиздат, М., 1983, 312 с. ↑25,26,27

[6] А. В. Забродин, В. В. Каратанов, В. В. Корнеев, В. К. Левин. «Массово-параллельные вычислительные системы на основе серийных микропроцессоров. Опыт создания и применения», *Труды Международной конференции «Параллельные вычисления и задачи управления»*, PACO'2001 (2–4 октября 2001, Москва, ИПУ РАН), с. 85–86. ↑25

[7] А. В. Забродин, В. К. Левин. «Опыт разработки параллельных вычислительных технологий. Создание и развитие семейства МВС», *Труды Всероссийской научной конференции «Высокопроизводительные вычисления и их приложения»* (30 октября–2 ноября 2000 года, Черноголовка), с. 3–8. ↑25

[8] А. О. Лацис. *Как построить и использовать суперкомпьютер*, Бестселлер, М., 2003, ISBN 5-98158-003-8, 240 с. ↑25

[9] М. А. Копытов, Г. М. Михайлов, Ю. П. Рогов. «Высокопроизводительный кластер вычислительного центра им. А. А. Дородницина РАН», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), МГУ, М., с. 60–62. ↑25

[10] С. М. Абрамов, А. И. Адамович, М. Р. Коваленко, А. Ф. Слепухин, Н. Н. Парамонов. «Кластерные системы семейства суперкомпьютеров «СКИФ»», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), МГУ, М., с. 147–151. ↑25

[11] А. А. Букатов, В. Н. Дацюк, О. В. Дацюк, Г. М. Хачкинаев. «Опыт создания высокопроизводительного кластера с использованием двух

коммуникационных сетей», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), МГУ, М., с. 110–112. ↑25

[12] М. П. Филамофитский. «Система X-COM: организация распределенных вычислений в сети Интернет», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), МГУ, М., с. 363–367. ↑25

[13] Ю. С. Затуливетер. «К глобальному компьютеру», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), МГУ, М., с. 186–190. ↑25

[14] Ю. Н. Галюк, В. И. Золотарев, В. П. Менонов. «GRID: отказоустойчивая схема многокластерных вычислений», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), МГУ, М., с. 87. ↑25

[15] R. Allen, K. Kennedy. *Optimizing Compilers for Modern Architectures: A Dependence-based Approach*, 1st ed., Morgan Kaufmann Publisher, 2001, ISBN 978-1558602861, 816 pp. ↑25,40,42,45

[16] П. М. Коуги. *Архитектура конвейерных ЭВМ*, Пер. с англ., Радио и связь, М., 1985, 358 с. ↑25,27

[17] К. Г. Самофалов, Г. М. Луцкий. *Основы теории многоуровневых конвейерных вычислительных систем*, Радио и связь, М., 1989, 272 с. ↑25

[18] И. А. Каляев, И. И. Левин, Е. А. Семерников, В. И. Шмойлов. *Реконфигурируемые мультиконвейерные вычислительные структуры*, 2-е, перераб. и доп. изд., ред. И. А. Каляев, Изд-во ЮНЦ РАН, Ростов-н/Д, 2009, 344 с. ↑25,37

[19] Б. А. Бабаян. «Уровень программирования и архитектурные принципы построения ЭВМ», Кибернетика и вычислительная техника, т. **2**, Наука, М., 1986, с. 18–27. ↑26

[20] B. A. Babayan. “Main Principles of E2K Architecture”, *Free Software Magazine*, **1**:2 (2002), pp. 13–26. URL ↑26,27

[21] Б. Бабаян, А. В. Бочаров, В. С. Волин, С. С. Гаврилов, А. С. Грошев, Ф. А. Груздов, М. В. Еремин, С. М. Зотов, А. Л. Плоткин, Л. Е. Пшеничников, Г. Г. Рябов, М. Л. Чудаков, В. С. Шевяков, *Многопроцессорные ЭВМ и методы их проектирования*, Перспективы развития вычислительной техники, т. **4**, Высшая школа, М., 1990, ISBN 5-06-000133-4, 144 с. ↑26

[22] P. C. Treleaven. “Parallel architecture overview”, *Parallel Computing*, **8**:1988, pp. 59–70. doi ↑26,27

[23] В. С. Подлазов. «Свойства мультикольцевых и гиперкубовых коммутаторов на произвольных перестановках», *Труды Международной конференции «Параллельные вычисления и задачи управления»*, PACO’2001 (2–4 октября 2001, Москва, ИПУ РАН), с. 152–164. ↑26

[24] А. А. Архангельская, В. А. Ершов, В. И. Нейман. *Автоматическая коммутация каналов связи*, Связь, М., 1970, 192 с. ↑26

[25] В. Э. Бенеш. *Математические основы теории телефонных сообщений*, Связь, М., 1968, 292 с. ↑26

[26] В. И. Кодачигов. *Электронная коммутация информационных каналов*, Изд-во Ростовского университета, Ростов-на-Дону, 1983, 208 с. ↑26

[27] А. В. Каляев. *Однородные коммутационные регистровые структуры*, Советское радио, М., 1978, 334 с. ↑26,37

[28] Г. Т. Артамонов. *Топология регулярных вычислительных сетей и сред*, Радио и связь, М., 1985, 192 с. ↑26

[29] Г. Т. Артамонов, В. Д. Тюрин. *Топология сетей ЭВМ и многопроцессорных систем*, Радио и связь, М., 1991, 248 с. ↑26

[30] В. В. Корнеев, В. Г. Хорошевский. *Архитектура вычислительных систем с программируемой структурой*, препринт ОВС-10, Институт математики СО АН СССР, Новосибирск, 1979, 48 с. ↑26,37

[31] В. В. Корнеев, В. Г. Хорошевский. *Структура и функциональная организация вычислительных систем с программируемой структурой*, препринт ОВС-11, Институт математики СО АН СССР, Новосибирск, 1979, 48 с. ↑26,37

[32] Sh. Akhter, R. Jason. *Multi-Core Programming. Increasing Performance Through Software Multi-threading*, Intel-Press, USA, 2006, 336 pp. ↑26

[33] В. Ю. Волконский, А. К. Ким. «Развитие идей параллелизма в архитектуре вычислительных комплексов серии «Эльбрус»», *Труды четвертой международной конференции «Параллельные вычисления и задачи управления»*, PACO'2008 (Москва, 27–29 октября 2008), с. 42–66. ↑27

[34] С. А. Орлов, Б. Я. Цилькер. *Организация ЭВМ и систем*, фундаментальный курс по архитектуре и структуре современных компьютерных средств, Питер,, Санкт-Петербург, 2015., ISBN 978-5-496-01145-7, 685 с. ↑27

[35] А. И. Солонина, Д. А. Улахович, Л. А. Яковлев. *Алгоритмы и процессоры цифровой обработки сигналов*, БХВ- Петербург, СПб., 2002, ISBN 5-94157-065-1, 454 с. ↑27

[36] В. В. Воеводин. *Математические модели и методы в параллельных процессах*, Наука, Главная редакция физико-математической литературы, М., 1986, 296 с. ↑27,31,51,58

[37] Б. А. Головкин. *Параллельные вычислительные системы*, Наука, Главная редакция физико-математической литературы, М., 1980, 518 с. ↑27

[38] В. В. Корнеев. *Параллельные вычислительные системы*, Нолидж, М., 1999, ISBN 5-89251-065-4, 320 с. ↑27

[39] Д. Ивенс (ред.). *Системы параллельной обработки*, Мир, М., 1985, 413 с. ↑27

[40] Р. У. Хокни, К. Р. Джесхоуп. *Параллельные ЭВМ: Архитектура, программирование и алгоритмы*, Радио и связь, М., 1986, 392 с. ↑27

[41] G. S. Almasi, A. Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Series in Computer Science and Engineering, Subsequent edition, Benjamin-Cummings Pub Co, 1993, ISBN 978-0805304435, 689 pp. ↑27,39

[42] Е. Валях. *Последовательно-параллельные вычисления*, пер. И. А. Николаев, А. М. Степанов, Мир, М., 1985, 456 с. ↑27

[43] Кун Сунь Юань. *Матричные процессоры на СБИС*, Мир, М., 1991, ISBN 5-03-001857-3, 672 с. ↑27

[44] В. Н. Белецкий. *Многопроцессорные и параллельные структуры с организацией асинхронных вычислений*, Наукова думка, Киев, 1988, ISBN 5-12-009328-0, 240 с. ↑27

[45] Н. Н. Миренков. *Параллельное программирование для многомодульных вычислительных систем*, Радио и связь, М., 1989, ISBN 5-256-00196-5, 320 с. ↑27

[46] J. Mukundan, H. Hunter, K.-H. Kim, J. Stuecheli, J. F. Martínez. "Understanding and Mitigating Refresh Overheads in High-Density DDR4 DRAM Systems", *ACM SIGARCH Computer Architecture News*, **43**:3 (2013). doi ↑27

[47] А. В. Линев, Д. К. Боголюбов, С. И. Бастраков, *Технологии параллельного программирования для процессоров новых архитектур*, Учебник, Суперкомпьютерное образование, ред. В. П. Гергель, Изд-во Московского ун-та, М., 2010, ISBN 978-5-211-05962-7, 152 с. ↑28

[48] A. Olofsson. *Epiphany-V: A 1024-core 64-bit RISC processor*, 2016, 15 pp. arXiv:1610.01832 URL ↑28,56

[49] А. А. Адамов, П. В. Павлухин, Д. В. Биконов, А. Л. Эйсымонт, Л. К. Эйсымонт. «Альтернативные современным GPGPU перспективные универсальные и специализированные процессоры-ускорители», *Вопросы кибербезопасности*, 2019, №4, с. 13–21. doi ↑28,56

[50] М. Кузьминский. «Power9 — процессоры для больших данных», *Открытые системы. СУБД*, 2017, №03 (Дата обращения 1.07.2020). URL https://www.osp.ru/os/2017/03/13052698 ↑28

[51] I. Grasso, P. Radojkovic, N. Rajovic, I. Gelado, and A. Ramírez.. "Energy efficient HPC on embedded SoCs: Optimization techniques for Mali GPU.", *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. doi ↑28

[52] Д. Харрис, С. Харрис. *Цифровая схемотехника и архитектура компьютера*, ДМК Пресс, 2018, ISBN 978-5-97060-570-7, 792 с. ↑28

[53] С. С. Андреев, С. А. Дбар, Ю. А. Климов, А. О. Лацис, Е. А. Плоткина. «Квантовая модель вычислений глазами классического программиста», *Препринты ИПМ им. М. В. Келдыша*, 2018, 178, 30 с. doi ↑28

[54] С. А. Степаненко. «Фотонный компьютер: структура и алгоритмы, оценки параметров», *Фотоника*, 2017, №7, с. 72–83. doi ↑28,38

[55] Е. Зуев. «Редкая профессия», *PC Magazine/Russian Edition*, 1997, №5 (75), Спецвыпуск; *Редкая профессия*, ДМК-Пресс, 2014, ISBN 978-5-94074-812-0. ↑29

[56] В. А. Вальковский. *Распараллеливание алгоритмов и программ: структурный подход*, Радио и связь, М., 1989, ISBN 978-5-256-00195-7, 176 с. ↑29,36,54

[57] M. R. Haghighat, C. D. Polychronopoulos. “Symbolic analysis for parallelizing compilers”, *ACM Transactions on Programming Languages and Systems*, **18**:4 (1995), pp. 477–518. ↑29,49

[58] Э. А. Трахтенгерц. *Программное обеспечение параллельных процессов*, Наука, М., 1987. ↑29

[59] А. В. Бабичев, В. Г. Лебедев. «Распараллеливание программных циклов», *Программирование*, **9**:5 (1983), с. 52–63. ↑29

[60] С. Абрамов, А. Адамович, М. Коваленко. «Т-система: среда программирования с поддержкой автоматического динамического распараллеливания на платформе «IP-сеть UNIX-компьютеров»», 4-ая Международная Российско-Индийская выставка-семинар «Математическое моделирование и визуализация» (15–25 сентября 1997 г., Москва). ↑31

[61] М. Е. Балашов, В. Д. Горячев, Д. С. Рыков, О. С. Рыкова, Е. М. Смирнов, Н. Е. Смирнова, С. А. Якубов. «Препроцессор для солвера CCFDD сетевой ИВС», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), Изд-во МГУ, М., с. 62–64. ↑31

[62] С. Л. Головков, А. Г. Смирнов В. К. Рубин. «Монитор поддержки параллельных научно-технических задач в сети Интернет», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), Изд-во МГУ, М., с. 79–82. ↑31

[63] А. С. Игумнов. «Открытая платформа отладки параллельных программ», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), Изд-во МГУ, М., с. 92–94. ↑31

[64] В. П. Гергель, А. Н. Свистунов. «Разработка интегрированной среды высокопроизводительных вычислений для кластера Нижегородского университета», Международный научно-практический семинар и молодежная школа «Высокопроизводительные параллельные вычисления на кластерных системах» (22–26 ноября 2005, ННГУ им. Н.И. Лобачевского), с. 51–54. ↑31

[65] А. И. Ильюшин, В. Н. Клепиков, А. В. Выродов, А. С. Бабанин, С. А. Будихин, К. В. Таченников. ««Подкачка» объектов как средство повышения пропускной способности МВС», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003, Нижний Новгород), Изд-во МГУ, М., с. 345–347. ↑31

[66] В. А. Евстигнеев, И. А. Мирзуитова. *Анализ циклов: выбор кандидатов на распараллеливание*, Препринт № 58, ИСИ РАН, Новосибирск, 1999, 49 с. ↑31,36

[67] В. А. Евстигнеев. «Некоторые особенности программного обеспечения ЭВМ с длинным командным словом (Обзор)», *Программирование*, **17**:2 (1991), с. 69–80. ↑31

[68] A. W. Lim, G. I. Cheong, M. S. Lam. "An affine partitioning algorithm to maximize parallelism and minimize communication", *ICS'99: Proceedings of the 13th international conference on Supercomputing* (Rhodes Greece, June, 1999), 1999, ISBN 978-1-58113-164-2, pp. 228–237. ↑31

[69] A. W. Lim, M. S. Lam. "Cache Optimizations With Affine Partitioning", *Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing, Portsmouth, Virginia, March, 2001.*, 2001, 14 pp. ↑31

[70] A. W. Lim, M. S. Lam. "Maximizing parallelism and minimizing synchronization with affine partitions", *Parallel Computing*, **24**:3–4 (1998), pp. 445–475. ↑31

[71] А. В. Фролов. «Автоматизация преобразований ФОРТРАН-программ», *Тезисы докладов всероссийской научной конференции «Фундаментальные и прикладные аспекты разработки больших распределенных программных комплексов»* (Новороссийск, 21–26 сентября 1998), МГУ, М., 1998. ↑32,51

[72] А. В. Фролов. «Инструментальная система для распараллеливания Фортран-программ как пример использования Интернет-технологий в программировании», *Тезисы докладов всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 24–29 сентября 2001), МГУ, М., 2001, с. 221–222. ↑32

[73] А. В. Фролов. «Нахождение и использование ориентированных разрезов реальных графов алгоритмов», *Программирование*, **23**:4 (1997), с. 71–80. ↑32

[74] В. В. Коваль. «Современные методы трансформации программ», *Компьютерное моделирование. Вычислительные технологии*, ЦВВР, Ростов-на-Дону, 2003, с. 41–58. ↑32

[75] В. В. Луговой. «Методы реализации внутреннего представления программ в CASE-системе распараллеливания программ», *Компьютерное моделирование. Вычислительные технологии*, 2003, с. 91–99. ↑32

[76] А. А. Букатов, В. В. Коваль. «Методы реализации трансформационной машины многоцелевой системы трансформаций программ», *Информационные технологии*, 2004, №3, с. 24–30. ↑32

[77] А. А. Букатов, В. В. Луговой. «Инкрементальная модификация структур данных внутреннего представления программ в системе распараллеливания программ», *Искусственный интеллект*, 2003, №3, с. 15–22. ↑32

[78] Б. Я. Штейнберг. «Вершины области изменения параметров циклов и информационная независимость», *Тезисы докладов I-й Всесоюзной конференции «Однородные вычислительные среды и систолические структуры»*. Т. 3 (17–20 апреля 1990, г. Львов), с. 112–116. ↑32

[79] Б. Я. Штейнберг. «Распараллеливание с анализом области изменения параметров циклов и с анализом внешних переменных в индексных выражениях», *Разработка системного и прикладного программного обеспечения МВК ПС-2000/2100, ПС-3000/3100*, Тезисы докладов Всесоюзного научно-технического семинара (г. Калинин), М., 1990, с. 5–6. ↑32

[80] Б. Я. Штейнберг. «Оптимальные параллельные переразмещения двумерных массивов», *Программирование*, **19**:6 (1993), с. 81–87. ✻ ↑32,56,57

[81] Б. Я. Штейнберг. «Бесконфликтные размещения массивов при параллельных вычислениях», *Кибернетика и системный анализ*, 1999, №1, с. 166–178. ↑32,56

[82] Б. Я. Штейнберг. «Оптимальные параллельные переразмещения многомерных массивов при параллельных вычислениях», *Сборник трудов Международной научно-технической конференции «Интеллектуальные многопроцессорные системы»* (1–5 сентября, 1999, Таганрог, Россия), с. 151–155. ↑32,56,57

[83] Б. Я. Штейнберг. «Разбиение циклов для исполнения на суперкомпьютере с архитектурой перестраиваемого конвейера», *Искусственный интеллект*, 2002, №3, с. 331–338. ↑32

[84] Б. Я. Штейнберг. «Подстановка и переименование индексных переменных в многомерных циклах», *Известия вузов. Северо-Кавказский регион. Естественные науки*, 2002, Юбилейный выпуск, с. 94–99. ↑32,51

[85] Б. Я. Штейнберг. «Подстановка и переименование в многомерных циклах при автоматическом распараллеливании», *Материалы Международной научно-технической конференции «СуперЭВМ и многопроцессорные вычислительные системы»*, МВС'2002 (Россия, Таганрог, 26–30 июня 2002), с. 161–164. ↑32

[86] Б. Я. Штейнберг. «Распараллеливание рекуррентных программных циклов», *Информационные технологии*, 2004, №4, с. 16–23. ↑32,58

[87] Б. Я. Штейнберг. *Математические методы распараллеливания рекуррентных программных циклов для суперкомпьютеров с параллельной памятью*, Изд-во Ростовского университета, Ростов-на-Дону, 2004, 192 с. ↑32,40,42,45,58

[88] Б. Я. Штейнберг, Д. В. Макошенко, Д. Н. Черданцев, А. М. Шульженко. «Внутреннее представление в открытой распараллеливающей системе», *Искусственный интеллект*, 2003, №3, с. 89–96. ↑32

[89] Б. Я. Штейнберг, Д. Н. Черданцев, С. А. Науменко, А. Э. Бутов, В. В. Петренко. «Преобразования программ для открытой распаралле-

ливающей системы», *Искусственный интеллект*, 2003, №3, с. 97–104. ↑32

[90] Б. Я. Штейнберг, М. В. Напрасникова. «Минимальное множество контрольных дуг при тестировании программных модулей», *Известия ВУЗов. Северо-Кавказский регион. Естественные науки*, 2003, №4, с. 15–18. ↑32,48

[91] Б. Я. Штейнберг, О. Э. Арутюнян, А. Э. Бутов, К. Ю. Гуфан, Р. Морылев, С. А. Науменко, В. В. Петренко, А. Тузаев, Д. Н. Черданцев, М. В. Шилов, Р. Б. Штейнберг, А. М. Шульженко. «Обучающая распараллеливанию программа на основе ОРС», *Труды научно-методической конференции «Современные информационные технологии в образовании: Южный федеральный округ»* (Ростов-на-Дону, 12–15 мая 2004), с. 248–250. ↑32,51

[92] Б. Я. Штейнберг, М. В. Напрасникова, З. Я. Нис. «Тестирование преобразований Открытой распараллеливающей системы», *Искусственный интеллект*, 2004, №3, с. 257–264. ↑32,48

[93] Б. Я. Штейнберг. «Открытая распараллеливающая система», *Открытые системы. СУБД*, 2007, №9, с. 36–41. URL ↑32,51

[94] Б. Я. Штейнберг. «Параллельное умножение разреженной матрицы на вектор», *Известия вузов. Северо-Кавказский регион. Естественные науки*, 2005, Специальный выпуск, с. 122–124. ↑32

[95] Б. Я. Штейнберг, З. Я. Нис, В. В. Петренко, Д. Н. Черданцев, Р. Б. Штейнберг, А. М. Шульженко. «Состояние и возможности открытой распараллеливающей системы (лето 2006 г.)», *Труды семинара «Наукоемкое программное обеспечение» в рамках шестой международной конференции памяти академика А. П. Ершова «Перспективы систем информатики»* (Новосибирск, Академгородок, 28–29 июня 2006), с. 122–125. ↑32,51

[96] Б. Я. Штейнберг, З. Я. Нис, В. В. Петренко, Д. Н. Черданцев, Р. Б. Штейнберг, А. М. Шульженко. «Открытая распараллеливающая система 2006 г.», *Труды III международной конференции «Параллельные вычисления и задачи управления»*, PACO'2006 (2–4 октября 2006, ИПУ РАН, Москва), с. 526–541. ↑32,51

[97] Б. Я. Штейнберг, Р. И. Морылев. «Распараллеливание программ с помощью Открытой распараллеливающей системы», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 18-22 сентября 2007), Изд-во МГУ, М.. ↑32,47,51

[98] А. М. Шульженко. «Автоматическое определение циклов ParDo в программе», *Известия вузов. Северо-Кавказский регион. Естественные науки*, 2005, №S11, с. 77–87. ↑32,51

[99] Р. Б. Штейнберг. «Вычисление задержки в стартах конвейеров для суперкомпьютеров со структурно процедурной организацией вычислений», *Искусственный интеллект*, 2003, №4, с. 105–112. ↑32,37

[100] О. Б. Штейнберг. «Автоматическое распараллеливание рекуррентных циклов с нерегулярным вычислением суперпозиций», *Труды четвертой международной конференции «Параллельные вычисления и задачи управления»*, PACO'2008 (Москва, 27–29 октября 2008). ↑32,47,58

[101] А. Н. Андрианов, К. Н. Ефимкин, И. Б. Задыхайло. «Непроцедурный язык для решения задач математической физики», *Программирование*, **17**:2 (1991), с. 80–94. ↑32

[102] А. Н. Андрианов. «Использование языка Норма для решения вычислительных задач на нерегулярных сетках», *Труды Всероссийской научной конференции «Фундаментальные и прикладные аспекты разработки больших распределенных программных комплексов»* (Новороссийск, 21–26 сентября 1998), Изд-во МГУ, М., с. 120–123. ↑32

[103] А. Н. Андрианов, Т. П. Баранова, А. Б. Бугеря, К. Н. Ефимкин. «Непроцедурный язык НОРМА и методы его трансляции для параллельных архитектур», *Известия высших учебных заведений. Северо-Кавказский регион. Технические науки*, 2017, №3 (195), с. 5–12. doi ↑32

[104] В. Ф. Алексахин, В. Н. Ильяков, Н. А. Коновалов, Н. В. Ковалева, В. А. Крюков, Н. В. Поддерюгина, Ю. Л. Сазанов. «Система автоматизации разработки параллельных программ для вычислительных кластеров и сетей (DVM-система)», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 23–28 сентября 2002), Изд-во МГУ, М., 2002, с. 272–273. ↑32,55

[105] В. Н. Коваленко И. В. Крюков В. А. Ильяков. «Анализ и предсказание эффективности выполнения DVM-программ на неоднородных сетях ЭВМ», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 22–27 сентября 2003), Изд-во МГУ, М., 2003, с. 181–182. ↑32,55

[106] В. А. Бахтин, Д. А. Захаров, А. А. Ермичев, В. А. Крюков. «Отладка параллельных программ в DVM-системе», *Электронные библиотеки*, **23**:4 (2020), с. 866–886. doi ↑32,33,55

[107] В. А. Бахтин, Д. А. Захаров, А. С. Колганов, В. А. Крюков, Н. В. Поддерюгина, М. Н. Притула. «Решение прикладных задач с использованием DVM-системы», *Вестн. ЮУрГУ. Сер. Выч. матем. информ.*, **8**:1 (2019), с. 89–106. doi ↑32,33,55

[108] В. А. Бахтин, Д. А. Захаров, А. А. Ермичев, В. А. Крюков, Н. В. Поддерюгина, М. Н. Притула. «Опыт решения прикладных задач, использующих нерегулярные сетки, с использованием DVM-системы», Короткие статьи и описания плакатов XIII Международной научной конференции «Параллельные вычислительные технологии», ПаВТ'2018, 2018, с. 241–252. ↑32,33,55

[109] В. А. Бахтин, Д. А. Захаров, В. А. Крюков, Н. В. Поддерюгина, М. Н. Притула. «Инкрементное распараллеливание программ с использованием DVM-системы», *Труды международной конференции*

«Суперкомпьютерные дни в России», Суперкомпьютерный консорциум университетов России, Российская академия наук, 2018, с. 991–993. ↑32,33,55

[110] В. А. Бахтин, О. Ф. Жукова, Н. А. Катаев, А. С. Колганов, В. А. Крюков, М. Ю. Кузнецов, Н. В. Поддерюгина, М. Н. Притула, О. А. Савицкая, А. А. Смирнов. «Распараллеливание программных комплексов. Проблемы и перспективы», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (17-22 сентября 2018 г., г. Новороссийск), ИПМ им. М. В. Келдыша, М., 2018, ISBN 978-5-98354-046-0, с. 63–72. ↑32,33,55

[111] А. А. Ермичев, В. А. Крюков. «Развитие метода сравнительной отладки DVMH-программ», *Труды XIX Всероссийской научной конференции «Научный сервис в сети Интернет»* (18–23 сентября 2017 г., г. Новороссийск), ИПМ им. М. В. Келдыша, М., 2017, ISBN 978-5-98354-037-8, с. 150–156. ↑32,33,55

[112] А. С. Колганов, В. А. Крюков, К. О. Шохин. «Распараллеливание циклов с регулярными зависимостями по данным на кластеры с графическими процессорами», *Тезисы докладов научной конференции «Ломоносовские чтения»* (Москва, 17–26 апреля 2017 года), ООО «МАКС Пресс», М., 2017, ISBN 978-5-89407-572-3, с. 29–30. ↑32,33,55

[113] В. А. Бахтин, О. Ф. Жукова, А. С. Колганов, Н. Н. Королев, В. А. Крюков, М. Ю. Кузнецов, Н. В. Поддерюгина, М. Н. Притула, О. А. Савицкая, А. А. Смирнов, Н. А. Катаев. «Инкрементное распараллеливание для кластеров в системе САПФОР», *Труды XIX Всероссийской научной конференции «Научный сервис в сети Интернет»* (18–23 сентября 2017 г., г. Новороссийск), ИПМ им. М. В. Келдыша, М., 2017, ISBN 978-5-98354-037-8, с. 48–52. ↑32,33,55

[114] В. Ф. Алексахин, В. А. Бахтин, Д. А. Захаров, А. С. Колганов, А. В. Королев, В. А. Крюков, Н. В. Поддерюгина, М. Н. Притула. «Опыт решения прикладных задач с использованием DVM-системы», *Труды международной конференции «Суперкомпьютерные дни в России»*, 2017, с. 650–661. ↑32,33,55

[115] V. Bakhtin, A. Kolganov, V. Krukov, N. Podderyugina, M. Pritula, O. Savitskaya. “An extension of the DVM-system to solve problems with intensive irregular memory access”, GraphHPC 2017 Proceedings of the 4th GraphHPC Conference on Large-Scale Grap Processing Using HPC Systems, CEUR Workshop Proceedings, vol. **1981**, 2017, pp. 25–30. ↑32,33,55

[116] В. А. Бахтин, А. С. Колганов, В. А. Крюков, Н. В. Поддерюгина, М. Н. Притула. «Расширение возможностей DVM-системы для решения задач, использующих нерегулярные сетки», *Труды международной конференции «Суперкомпьютерные дни в России»* (26–27 сентября 2016 г., г. Москва), Изд-во МГУ, М., 2016, с. 596–603. ↑32,33,55

[117] В. А. Бахтин, В. А. Крюков, Н. В. Поддерюгина, М. Н. Притула. «Реализация параллельного ввода-вывода в DVM-системе», *Труды международной конференции «Суперкомпьютерные дни в России»* (26–27 сентября 2016 г., г. Москва), Изд-во МГУ, М., 2016, с. 615–625. URL ↑32,33,55

[118] В. А. Бахтин, А. С. Колганов, В. А. Крюков, Н. В. Поддерюгина, С. В. Поляков, М. Н. Притула. «Расширение DVMH-модели для работы с нерегулярными сетками», *Труды международной научной конференции «Параллельные вычислительные технологии»*, ПаВТ'2016, Издательский центр ЮУрГУ, Челябинск, 2016, с. 757. URL ↑32,33,55

[119] В. А. Бахтин, О. Ф. Жукова, Н. А. Катаев, А. С. Колганов, В. А. Крюков, Н. В. Поддерюгина, М. Н. Притула, О. А. Савицкая, А. А. Смирнов. «Автоматизация распараллеливания программных комплексов», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (19–24 сентября 2016 г., г. Новороссийск), ИПМ им. М.В. Келдыша РАН, М., 2016, с. 76–85. URL ↑32,33,55

[120] V. Krukov. "Automation of Programming at the M. V. Keldysh Institute of Applied Mathematics, Russian Academy of Sciences (KIAM RAS)", *Proceedings 3rd International Conference on Computer Technology in Russia and in the Former Soviet Union*, SoRuCom 2014 (13–17 Oct. 2014, Kazan, Russia), 2015, ISBN 978-1-4799-1799-0, pp. 127–130. doi ↑32,33,55

[121] В. Ф. Алексахин, В. А. Бахтин, О. Ф. Жукова, А. С. Колганов, В. А. Крюков, И. П. Островская, Н. В. Поддерюгина, М. Н. Притула, О. А. Савицкая. «Распараллеливание на языке Fortran-DVMH для сопроцессора Intel Xeon Phi тестов NAS NPB3.3.1», *Труды международной научной конференции «Параллельные вычислительные технологии»*, ПаВТ'2015, Издательский центр ЮУрГУ, Челябинск, 2015, с. 19–30. URL ↑32,33,55

[122] В. А. Бахтин, О. Ф. Жукова, Н. А. Катаев, А. С. Колганов, Н. В. Поддерюгина, М. Н. Притула, О. А. Савицкая, А. А. Смирнов, В. А. Крюков. «Использование интернета для обучения параллельному программированию», *Труды XVII Всероссийской научной конференции «Научный сервис в сети Интернет»* (21–26 сентября 2015 г., г. Новороссийск), ИПМ им. М. В. Келдыша, 2015, с. 26–33. URL ↑32,33,55

[123] В. Ф. Алексахин, В. А. Бахтин, О. Ф. Жукова, А. С. Колганов, В. А. Крюков, Н. В. Поддерюгина, М. Н. Притула, О. А. Савицкая, А. В. Шуберт. «Распараллеливание на графические процессоры тестов NAS NPB 3.3.1 на языке FORTRAN-DVMH», *Вестник Уфимского государственного авиационного технического университета*, **19**:1 (2015), с. 240–250. ↑32,33,55

[124] В. Ф. Алексахин, В. А. Бахтин, О. Ф. Жукова, А. С. Колганов, В. А. Крюков, И. П. Островская, Н. В. Поддерюгина, М. Н. Притула, О. А. Савицкая. «Распараллеливание тестов NAS NPB для сопроцессора Intel Xeon Phi на языке Fortran-DVMH», *Вестн. ЮУрГУ. Сер. Выч. матем. информ.*, **4**:4 (2015), с. 48–63. doi ↑32,33,55

[125] В. А. Бахтин, Н. А. Коновалов, В. А. Крюков. «Расширение языка OpenMP Fortran для программирования GRID-приложений», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 23–28 сентября 2002), Изд-во МГУ, М., с. 273. ↑33

[126] V. Kasyanov, E. Kasyanova. “Methods and tools of parallel programming”, International Conference Mathematical and Information Technologies (Vrnjacka Banja, Serbia–Budva, Montenegro, August 28–September 5, 2016), CEUR Workshop Proceedings, vol. **1839**, 2017, pp. 141–154. URL ↑33

[127] В. П. Гергель, Р. Г. Стронгин. *Основы параллельных вычислений для многопроцессорных вычислительных систем*, 2 изд., ННГУ, Н. Новгород, 2003, ISBN 5-85746-602-4, 184 с. ↑33,36

[128] Ж. М. Абу-Халил, С. А. Гуда, Б. Я. Штейнберг. «Перенос параллельных программ с сохранением эффективности», *Открытые системы. СУБД*, 2015, №4, с. 18–19. URL ↑33,61

[129] Ж. М. Абу-Халил, С. А. Гуда, Б. Я. Штейнберг. «О высокопроизводительной переносимости программ», Труды научной школы И. Б. Симоненко, т. **2**, Изд-во Южного федерального университета, Ростов-на-Дону, 2015, ISBN 978-5-9275-1607-0, с. 26–33. URL ↑33

[130] K. Goto, R. A. van de Geijn. “Anatomy of high-performance matrix multiplication”, *ACM Trans. Math. Softw.*, **34**:3 (2008), 12, 25 pp. doi ↑34,50,52

[131] Р. Бэбб, Дж. Мак-Гроу, Т. Аксельрод и др.. *Программирование на параллельных вычислительных системах*, ред. Р. Бэбб, Ю. Г. Дадаев, Мир, М., 1991, ISBN 5-03-001527-2, 376 с. ↑34

[132] В. А. Вальковский. «Параллельное выполнение циклов. Метод пирамид», *Кибернетика*, 1983, №5, с. 51–55. ↑35,51

[133] В. А. Вальковский. «Параллельное выполнение циклов. Метод параллелепипедов», *Кибернетика*, 1982, №2, с. 51–62. ↑35,51

[134] D. Kulkurni, M. Stumm. *Loop and data transformations: a tutorial*, Technical Report CSRI-337, Computer Systems Research Institute, University of Toronto, 1993, 53 pp. ↑35

[135] K. E. Schauser. *Compiling dataflow into threads: efficient compiler-controlled multithreading for lenient parallel languages*, Technical Report No UCB/CSD-91-644, University of California, Berkley, 1991, 71 pp. URL ↑35

[136] N. Saboo, A. K. Singla, J. M. Under, L. V. Kale. “Emulating petaflops machines and blue gene”, *Proceedings 15th International Parallel and Distributed Processing Symposium*, IPDPS 2001 (23–27 April 2001, San Francisco, CA, USA), 2001, ISBN 0-7695-0990-8, pp. 2084–2091. doi ↑35

[137] С. В. Торчигин. «Особенности отладки прикладных и системных задач на вычислительной системе с автоматическим распределением ресурсов», *Тезисы докладов Международной научной конференции*

«Интеллектуальные и многопроцессорные системы», Изд-во ТРТУ, Таганрог, 2003, с. 55–57. ↑35

[138] А. Е. Ширай. «Системная поддержка вычислений в комплексе с автоматическим распределением ресурсов», *Тезисы докладов Международной научной конференции «Интеллектуальные и многопроцессорные системы»*, Изд-во ТРТУ, Таганрог, 2003, с. 54–55. ↑35

[139] В. С. Бурцев. «Новые подходы к оценке качества вычислительных средств», *Параллелизм вычислительных процессов и развитие архитектуры суперЭВМ*, Нефть и газ, М., 1997, с. 28–40. ↑35

[140] В. Е. Котов, Л. А. Черкасова. «Сетевой подход к описанию семантики параллельных систем и процессов», Кибернетика и вычислительная техника, т. **2**, Наука, М., 1986, с. 75–94. ↑35

[141] В. Е. Котов. *Сети Петри*, Наука. Главная редакция физико-математической литературы, М., 1984, 159 с. ↑35

[142] С. Д. Бахтеяров. *Язык программирования ОККАМ*, МНИИПУ, М., 1989, 86 с. ↑35

[143] А. С. Антонов. *Параллельное программирование с использованием технологии MPI*, Изд-во МГУ, М., 2004, ISBN 5-211-04907-1, 77 с. URL ↑35

[144] А. А. Букатов, В. Н. Дацюк, А. И. Жегуло. *Программирование многопроцессорных вычислительных систем*, Изд-во ООО «ЦВВР», Ростов-на-Дону, 2003, ISBN 5-94153-062-5, 208 с. URL ↑35

[145] С. Немнюгин, О. Стесик. *Параллельное программирование для многопроцессорных вычислительных систем*, БХВ-Петербург, СПб., 2002, ISBN 5-94157-188-7, 400 с. ↑35

[146] В. Н. Касьянов. *Оптимизирующие преобразования программ*, Наука, М., 1988, ISBN 5-02-013778-2, 338 с. ↑35,40

[147] К. Касперский. *Техника оптимизации программ. Эффективное использование памяти*, БХВ-Петербург, СПб., 2003, ISBN 5-94157-232-8, 464 с. ↑35,40,42

[148] L. Lamport. “The Coordinate Method for the parallel execution of DO loops”, Sagamore Computer Conference on Parallel Processing, 1973, pp. 1–12. ↑36

[149] L. Lamport. “The parallel execution of DO loops”, *Commun. ACM*, **17**:2 (1974), pp. 83–93. doi ↑36,51

[150] В. А. Евстигнеев, В. Н. Касьянов. «Оптимизирующие преобразования в распараллеливающих компиляторах», *Программирование*, **22**:6 (1996), с. 12–26. ✻ ↑36,40

[151] А. П. Черняев. «Системы программирования для высокопроизводительных ЭВМ», Итоги науки и техники. Вычислительные науки, т. **3**, ВИНИТИ АН СССР, М., 1990, с. 1–141. ↑36

[152] А. П. Черняев. «Программные системы векторизации и распараллеливания ФОРТРАН-программ для некоторых векторно-конвейерных ЭВМ (обзор)», *Программирование*, **17**:2 (1991), с. 53–68. ↑36

[153] Ю. А. Французов. «Обзор методов распараллеливания кода и программной конвейеризации», *Программирование*, **18**:3 (1992), с. 16–37. ↑36,40

[154] Р. Аллен, К. Кеннеди. «Автоматическая трансляция Фортран-программ в векторную форму», *Векторизация программ: теория, методы, реализация*, Мир, М., 1991, с. 77–140. ↑36

[155] В. А. Серебряков. «Циклическая программная конвейеризация и трансляция DO циклов для сильносвязанных многопроцессорных систем», *Программирование*, **18**:3 (1992), с. 54–60. ↑36

[156] U. Gupta, B. Reagen, L. Pentecost, M. Donato, Th. Tambe, A. M. Rush, Gu-Yeon Wei, D. Brooks. "MASR: A modular accelerator for Sparse RNNs", 28th International Conference on Parallel Architectures and Compilation Techniques, PACT 2019 (September 23-26, 2019, Seattle, WA, USA), pp. 1–14. ↑36

[157] Xiao Dong, Lei Liu, Peng Zhao, Guangli Li, Jiansong Li, Xueying Wang, Xiaobing Feng. "Acorns: a framework for accelerating deep neural networks with input sparsity", 28th International Conference on Parallel Architectures and Compilation Techniques, PACT 2019 (September 23-26, 2019, Seattle, WA, USA), pp. 178–191. ↑36

[158] D.-M. Loroch, F.-J. Pfreundt, N. Wehn, J. Keuper. "Sparsity in Deep Neural Networks — An Empirical Investigation with TensorQuant", Joint European Conference on Machine Learning and Knowledge Discovery in Databases (27 Aug. 2018), Communications in Computer and Information Science, vol. **967**, Springer, Cham, ISBN 978-3-030-14879-9, pp. 5–20. ↑36

[159] A. Guha, N. Vedula, A. Shriraman. "Deepframe: A Profile-Driven Compiler for Spatial Hardware Accelerators", 28th International Conference on Parallel Architectures and Compilation Techniques, PACT 2019 (September 23-26, 2019, Seattle, WA, USA), pp. 68–81. ↑37

[160] Ю. П. Кондратенко, В. В. Мохор, С. А. Сидоренко. *Verilog-HDL для моделирования и синтеза цифровых электронных схем*, Учебное пособие, Издательство НГТУ, Николаев, 2002, 206 с. ↑37

[161] И. В. Хаханова. *Модели и методы системного проектирования вычислительных структур на кристаллах для цифровой обработки сигналов*, Диссертация на соискание ученой степени доктора технических наук, ХНУРЭ, Харьков, 2008, 334 с. ↑37

[162] А. М. Сергиенко. *VHDL для проектирования вычислительных устройств*, ЧП «Корнейчук», ООО «ТИД «ДС», К., 2003, ISBN 966-7599-32-9, 208 с. ↑37

[163] Р. Б. Штейнберг. «Использование решетчатых графов для исследования многоконвейерной модели вычислений», *Известия ВУЗов. Северо-Кавказский регион. Естественные науки*, 2009, №2, с. 16–18. ↑37

[164] Р. Б. Штейнберг. «Отображение гнезд циклов на многоконвейерную вычислительную архитектуру», *Программирование*, **36**:3 (2010), с. 69–80. ↑37

[165] А. В. Каляев. «Программирование виртуальных параллельных проблемно-ориентированных суперкомпьютеров в структуре универсальных суперкомпьютеров с массовым параллелизмом», *Труды международной научно-технической конференции «Интеллектуальные многопроцессорные системы»* (Таганрог, 1–5 сентября 1999), с. 27–39. ↑37,51

[166] А. В. Каляев, И. И. Левин. *Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений*, Янус-К, М., 2003, 380 с. ↑37,39

[167] И. И. Левин. «Модульно-наращиваемая многопроцессорная вычислительная система со структурно-процедурной организацией вычислений на основе ПЛИС-технологий», *Искусственный интеллект*, 2003, №4, с. 446–453. ↑37

[168] А. В. Каляев, А. Ю. Арцатбанов, И. И. Итенберг. «Принципы построения семейства высокопроизводительных ортогональных многопроцессорных вычислительных систем с программируемой архитектурой», Многопроцессорные вычислительные структуры, т. **13(22)**, Таганрог, 1991, с. 4–9. ↑37

[169] А. В. Каляев, И. И. Итенберг. «Организация мультисупертранспьютерных вычислительных систем на основе макрокомпьютеров с программируемой архитектурой», *Тезисы докладов IV Всесоюзного семинара «Распределенная обработка информации»* (Новосибирск, 1991), с. 4. ↑37

[170] А. В. Каляев. *Многопроцессорные системы с программируемой архитектурой*, Радио и связь, М., 1984, 240 с. ↑37

[171] В. В. Корнеев. «Программная настраиваемость аппаратной структуры», *Открытые системы. СУБД*, 2007, №10. URL ↑37

[172] В. В. Корнеев. *Архитектура вычислительных систем с программируемой структурой*, Наука, Новосибирск, 1985, 166 с. ↑37

[173] Э. В. Евреинов, Ю. Г. Косарев. *Однородные универсальные вычислительные системы высокой производительности*, Наука, Новосибирск, 1966. ↑37

[174] K. Bondalapati. *Modeling and mapping for dynamically reconfigurable hybrid architecture*, Ph.D. Thesis, University of Southern California, 2001, 194 pp. URL ↑37,38

[175] K. Bondalapati, V. K. Prasanna. “Reconfigurable computing systems”, *Proceedings of the IEEE*, **90**:7 (2002), pp. 1201–1217. doi ↑37,38

[176] K. Compton, S. Hauck. “Reconfigurable computing: a survey of systems and software”, *ACM Computing Surveys*, **34**:2 (2002), pp. 171–210. ↑37

[177] M. S. Jadzhak. “On a numerical algorithm of solving the cascade digital filtration problem”, *Journal of Automation and Information Sciences*, **36**:6 (2004), pp. 23–34. ↑37

[178] M. S. Yadzhak, M. I. Tyutyunnyk. “An optimal algorithm to solve digital filtering problem with the use of adaptive smoothing”, *Cybernetics and Systems Analysis*, **49**:3 (2013), pp. 449–456. ↑37

[179] A. V. Anisimov, M. S. Yadzhak. “Construction of optimal algorithms for mass computations in digital filtering problems”, *Cybernetics and Systems Analysis*, **44**:4 (2008), pp. 465–476. ↑37

[180] М. Г. Адигеев, Д. В. Дубров, С. А. Лазарева, Б. Я. Штейнберг. «Экспериментальный распараллеливающий компилятор на Супер-ЭВМ со структурной организацией вычислений», *Тезисы докладов всероссийской научной конференции «Фундаментальные и прикладные аспекты разработки больших распределенных программных комплексов»* (Новороссийск, 21–26 сентября 1998), Изд-во МГУ, М., 1998, с. 101–108. ↑38

[181] D. Dubrov, A. Roshal. “Generating pipeline integrated circuits using C2HDL converter”, East-West Design & Test Symposium (EWDTS 2013) (27-30 Sept. 2013, Rostov on Don, Russia), pp. 1–4. ↑38

[182] B. Ya. Steinberg, D. V. Dubrov, Yu. V. Mikhailuts, A. S. Roshal, R. B. Steinberg. “Automatic high-level programs mapping onto programmable architectures”, *Proceedings of the 13th International Conference on Parallel Computing Technologies* (August 31–September 4, 2015, Petrozavodsk, Russia), Lecture Notes in Computer Science, vol. **9251**, Springer, Cham, ISBN 978-3-319-21908-0, pp. 474–485. ↑38

[183] B. Ya. Steinberg, A. P. Bugliy, D. V. Dubrov, Yu. V. Mikhailuts, O. B. Steinberg, R. B. Steinberg. “A project of compiler for a processor with programmable accelerator”, 5th International Young Scientist Conference on Computational Science, YSC 2016 (26–28 October 2016, Krakow, Poland), *Procedia Computer Science*, **101** (2016), pp. 435–438. ↑38

[184] A. P. Bugliy, D. V. Dubrov, Y. V. Mikhailuts, B. Ya. Steinberg, R. B. Steinberg. “Developing a high-level language compiler for a computer with programmable architecture”, *CEE-SECR’16: Proceedings of the 12th Central and Eastern European Software Engineering* (October 2016, Moscow, Russia), ISBN 978-1-4503-4884-3, 1–6 pp. ↑38

[185] Y. Liu, B. Schmidt. “SWAPHI: Smith-Waterman protein database search on Xeon Phi coprocessors”, *ASAP 2014*. V. 1, 2014, pp. 184–185. ↑38,61

[186] Д. В. Дубров, А. С. Рошаль, Б. Я. Штейнберг, Р. Б. Штейнберг. «Автоматическое отображение программ на процессор с плисускорителем», *Вестн. ЮУрГУ. Сер. Выч. матем. информ.*, **3**:2 (2014), с. 117–121. ↑38

[187] K. Bondalapati, V. K. Prasanna. “Loop pipelining and optimization for run time reconfiguration”, *IPDPS 2000: Parallel and Distributed Processing*,

International Parallel and Distributed Processing Symposium, Lecture Notes in Computer Science, vol. **1800**, Springer, Berlin–Heidelberg, 2000, ISBN 978-3-540-67442-9, pp. 906–915. ↑38

[188] B. Y. Steinberg, A. P. Bagliy, Zh. M. Petrova, O. B. Steinberg. "Pipeline circuits to compute several expressions", *CEE-SECR'18: Proceedings of the 14th Central and Eastern European Software Engineering* (October 2018, Moscow, Russian Federation), ISBN 978-1-4503-6176-7, pp. 1–7. ↑38

[189] А. П. Баглий, Д. В. Дубров, Б. Я. Штейнберг, Р. Б. Штейнберг. «Повторное использование ресурсов при конвейерных вычислениях», *Труды XIX Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 18–23 сентября 2017), ИПМ им. М.В. Келдыша, М., 2017, с. 43–47. ↑38

[190] Б. Я. Штейнберг, А. П. Баглий, Д. В. Дубров, Ю. В. Михайлуц, О. Б. Штейнберг, Р. Б. Штейнберг. «Классификация циклов с одним оператором для выполнения на процессоре с программируемым ускорителем», *Программные системы: теория и приложения*, **8**:3, с. 189–218. ↑38,42

[191] В. И. Вьюкова, В. А. Галатенко, С. В. Самборский. «Использование ЦЛП подхода для совмещения программной конвейеризации внутреннего цикла с разверткой внешних циклов при компиляции гнезда вложенных циклов», *Труды четвертой международной конференции «Параллельные вычисления и задачи управления»*, PACO'2008 (Москва, 27–29 октября 2008), с. 1208–1220. ↑38

[192] С. С. Андреев, С. А. Дбар, А. О. Лацис, Е. А. Плоткина. «Система программирования Автокод HDL и опыт ее применения для схемной реализации численных методов в FPGA», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность»* (Новороссийск, 21–26 сентября 2009), Изд-во МГУ, М., 2009, с. 237. ↑38

[193] С. С. Андреев, А. А. Давыдов, С. А. Дбар, А. О. Лацис, Е. А. Плоткина. *Технологии разработки прикладного ПО для реконфигурируемых вычислительных структур (ПЛИС)*, 2010 (Дата обращения 1.07.2020), 27 с. ↑39

[194] D. J. Kuck, R. H. Kuhn, B. Leasure, M. Wolfe. "Dependence graphs and compiler optimizations", *POPL '81: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (Williamsburg, Va., Jan. 26–28), 1981, ISBN 978-0-89791-029-3, pp. 207–218. ↑39

[195] В. В. Петренко. «Внутреннее представление REPRISE распараллеливающей системы», *Труды четвертой международной конференции «Параллельные вычисления и задачи управления»*, PACO'2008 (27–29 октября 2008, Москва, Россия). ↑39

[196] Е. В. Алымова, А. П. Баглий, Д. В. Дубров, Р. А. Ибрагимов, Ю. В. Михайлуц, В. В. Петренко, Б. Я. Штейнберг, Р. Б. Штейнберг, В. А. Яковлев.

«О промежуточном представлении программ для автоматического создания конвейерных вычислителей», *Известия вузов. Северо-Кавказский регион. Технические науки*, 2017, №3, с. 22–29. ↑39

[197] В. Петренко, Е. Метелица, Р. Морылев, Б. Штейнберг. «Основанная на ОРС система обучения преобразованиям программ «Тренажер параллельного программиста»», *Труды Всероссийской научной конференции памяти А. Л. Фуксмана «Языки программирования и компиляторы»* (Ростов-на-Дону, 3–5 апреля 2017, Южный федеральный университет), Издательство Южного федерального университета, Ростов-на-Дону, 2017, с. 198–201. ↑39

[198] Е. Алымова, Б. Штейнберг, А. Баглий, Д. Дубров, В. Петренко, Р. Ибрагимов, Ю. Михайлуц, Р. Штейнберг. «Промежуточное представление программ ОРС для генерации схемы конвейерного вычислителя», *Труды Всероссийской научной конференции памяти А. Л. Фуксмана «Языки программирования и компиляторы»* (Ростов-на-Дону, 3–5 апреля 2017, Южный федеральный университет), Издательство Южного федерального университета, Ростов-на-Дону, 2017, с. 38–41. ↑39

[199] К. Гуфан, Р. Морылев, В. Петренко. «Визуализация в Открытой распараллеливающей системе», *Труды Всероссийской научной конференции «Научный сервис в сети Интернет»* (Новороссийск, 18–22 сентября 2006), с. 58–61. ↑39,45

[200] Е. Н. Акимова, Р. А. Гареев. «Аналитическое моделирование матрично-векторного произведения на многоядерных процессорах», *Вестн. ЮУрГУ. Сер. Выч. матем. информ.*, **9**:1 (2020), с. 69–82. ↑40,50,52

[201] R. Gareev, T. Grosser, M. Kruse. “High-performance generalized tensor operations: a compiler-oriented approach”, *ACM Transactions on Architecture and Code Optimization*, **15**:3 (2018), pp. 1–27, 34. ↑40,52

[202] S. Muchnick. *Advanced Compiler Design Implementation*, 1 ed., Morgan Kaufmann, 1997, ISBN 978-1558603202, 888 pp. ↑40

[203] А. Ахо, Дж. Ульман. *Теория синтаксического анализа, перевода и компиляции.* Т. 1: *Синтаксический анализ*, Мир, М., 1978, 616 с.; 243 с. ↑40

[204] А. Ахо, М. Лам, Р. Сети, Дж. Ульман. *Компиляторы. Принципы, технологии и инструментарий*, Пер. с англ., 2-е изд., Издательский дом «Вильямс», М., 2008, ISBN 978-5-8459-1932-8, 1184 с. ↑40

[205] M. Griebl. *Automatic parallelization of loop programs for distributed memory architectures*, Habilitation, Passau University, 2004, 207 pp. ↑40,45,55

[206] D. J. Kuck, R. H. Kuhn, B. Leasure, M. Wolfe. “Dependence graphs and compiler optimizations”, *POPL’81: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (Jan. 1981, Williamsburg, Va., USA), 1981, ISBN 978-0-89791-029-3, pp. 207–218. ↑40,45

[207] В. А. Евстигнеев, С. В. Спрогис. «Векторизация программ», *Векторизация программ: теория, методы, реализация*, Мир, М., 1991, ISBN 5-03-001943-X, с. 246–267. ↑40,42,43

[208] В. Н. Касьянов, В. А. Евстигнеев. *Графы в программировании: обработка, визуализация и применение*, БХВ-Петербург, СПб., 2003, ISBN 5-94157-184-4, 1104 с. ↑40

[209] Б. Я. Штейнберг, Е. В. Алымова, А. П. Баглий, С. А. Гуда, Е. Н. Кравченко, Р. И. Морылев, З. Я. Нис, В. В. Петренко, И. С. Скиба, В. Н. Шаповалов, О. Б. Штейнберг. «Особенности реализации распараллеливающих преобразований программ в ДВОР», *Труды международной конференции «Параллельные вычисления и задачи управления»*, PACO'2010 (Москва, 26–28 октября 2010, ИПУ РАН), с. 787–854. ↑40

[210] R. Cytron, J. Ferrante, B. Rosen, M. N. Wegman, F. K. Zadeck. "Efficiently computing static single assignment form and the control dependence graph", *ACM Transactions on Programming Languages and Systems*, **13**:4 (1991), pp. 451–490. ↑40

[211] R. Ballance, A. Maccabe, K. Ottenstein. "The program dependence web: a representation supporting control-, data-, and demanddriven interpretation of imperative languages", *ACM SIGPLAN Notices*, **25**:6 (1990), pp. 257–271. ↑40

[212] P. Havlak. "Construction of thinned gated single-assignment form", *LCPC 1993: Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, vol. **768**, Springer, Berlin–Heidelberg, 1993, ISBN 978-3-540-57659-4, pp. 477–499. ↑40

[213] Р. И. Морылев, В. Н. Шаповалов, Б. Я. Штейнберг. «Символьный анализ в диалоговом распараллеливании программ», *Информационные технологии*, 2013, №2, с. 33–36. ↑40,46

[214] Л. Р. Гервич, Е. Н. Кравченко, Б. Я. Штейнберг, М. В. Юрушкин. «Автоматизация распараллеливания программ с блочным размещением данных», *Сибирский журнал вычислительной математики*, **18**:1 (2015), с. 41–53 . % ↑41,50,52

[215] Ж. М. Абу-Халил, Р. И. Морылев, Б. Я. Штейнберг. «Параллельный алгоритм глобального выравнивания с оптимальным использованием памяти», *Современные проблемы науки и образования*, 2013, №1, с. 112–117. ↑41,51,52,61

[216] О. Б. Штейнберг. «Классификация программных циклов с одним оператором присваивания», *Известия ВУЗов. Северо-Кавказский регион. Естественные науки*, 2017, №1, с. 52–58. ↑41,42

[217] О. Б. Штейнберг. «Распараллеливание рекуррентных циклов с нерегулярным вычислением суперпозиций», *Известия ВУЗов. Северо-Кавказский регион. Естественные науки*, 2009, №2, с. 16–18. ↑41,42,47

[218] С. Е. Суховерхов, О. Б. Штейнберг. «Автоматическое распараллеливание рекуррентных циклов с проверкой устойчивости», *Информационные технологии*, 2010, №1, с. 40–45. ↑41,42,47

[219] О. Б. Штейнберг. «Минимизация количества временных массивов в задаче разбиения циклов», *Известия ВУЗов. Северо-Кавказский регион. Естественные науки*, 2011, №5, с. 31–35. ↑41,42

[220] O. B. Steinberg. «Circular shift of loop body — programme transformation, promoting parallelism», *Вестн. ЮУрГУ. Сер. Матем. моделирование и программирование*, **10**:3 (2017), с. 120–132. ↑41,42

[221] О. Б. Штейнберг, И. А. Ивлев. «Применение преобразования циклов «Retiming» с целью уменьшения количества используемых регистров», *Известия высших учебных заведений. Северо-Кавказский регион. Технические науки*, 2017, №3, с. 76–80. ↑41,42

[222] В. А. Савельев, Б. Я. Штейнберг. *Распараллеливание программ*, Изд-во Южного федерального университета, Ростов-на-Дону, 2008, ISBN 978-5-9275-0547-0, 192 с. ↑41

[223] В. В. Воеводин, Вл. В. Воеводин. *Параллельные вычисления*, БХВ-Петербург, СПб., 2002, ISBN 5-94157-160-7, 599 с. ↑42,51

[224] D. Liu, Z. Shao, M. Wang, M. Guo, J. Xue. “Optimal loop parallelization for maximizing iteration-level parallelism”, *CASES’09: Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems* (October 2009, Grenoble, France), 2009, ISBN 978-1-60558-626-7, pp. 67–76. ↑42

[225] D. Liu, Z. Shao, M. Wang, M. Guo, J. Xue. “Optimally maximizing iteration-level loop parallelism”, *IEEE Transactions on Parallel and Distributed Systems*, **23**:3 (2012), pp. 564–572. ↑42

[226] P. Feautrier. “Dataflow analysis for array and scalar references”, *International Journal of Parallel Programming*, **20**:1 (1991), pp. 23–53. ↑42,51

[227] В. Шампаров, А. Маркин. «Structure Splitting для компилятора для микропроцессоров Эльбрус», *CEE-SECR’19: Proceedings of the 14th Central and Eastern European Software Engineering Conference* (14–15 ноября 2019, Санкт-Петербург, Россия). ↑42,49

[228] P. P. Chang, W.-W. Hwu. “Inline function expansion for compiling C programs”, *ACM SIGPLAN Notices*, **24**:7 (1989), pp. 246–257. ↑43

[229] P. Zhao, J. N. Amaral. “To inline or not to inline? Enhanced inlining decisions”, *LCPC 2003: Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, vol. **2958**, Springer, Berlin–Heidelberg, 2003, ISBN 978-3-540-21199-0, pp. 405–419. ↑43

[230] P. Andersson. *Evaluation of inlining heuristics in industrial strength compilers for embedded systems*, Examensarbete, Uppsala Universitet, 2009, 38 pp. ↑43

[231] А. В. Климов, С. А. Романенко. «Суперкомпиляция: основные принципы

и базовые понятия», *Препринты ИПМ им. М. В. Келдыша*, 2018, 111, 36 с. ↑43

[232] Д. В. Макошенко. *Аналитическое предсказание времени исполнения программ и основанные на нем методы оптимизации*, Диссертация на соискание степени кандидата физико-математических наук по специальности 05.13.11: математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей, ИСИ СО РАН им. А. П. Ершова, Новосибирск, 2011, 122 с. ↑44

[233] Вл. В. Воеводин. «Статистические оценки возможности выявления параллельной структуры последовательных программ», *Программирование*, **16**:4 (1990), с. 44–54. ↑45

[234] Вл. В. Воеводин. «Теория и практика исследования параллелизма последовательных программ», *Программирование*, **18**:3 (1992), с. 38–54. ↑45

[235] С. А. Хожайнова. «Статистические данные о распараллеливаемости программ», *Тезисы докладов конференции «Смешанные вычисления и преобразования программ»* (Новосибирск, Институт систем информатики и вычислительный центр АН СССР, 27–29 ноября 1990), ВЦ СО РАН, Новосибирск, 1991, с. 237–242. ↑45

[236] Р. Н. Арапбаев. *Анализ зависимостей по данным: тесты на зависимость и стратегии тестирования*, Диссертация на соискание ученой степени кандидата физико-математических наук, ИСИ СО РАН, Новосибирск, 2008, 116 с. ↑45

[237] W. Pugh. “The Omega test: a fast and practical integer programming algorithm for dependence analysis”, *Supercomputing’91: Proceedings of the 1991 ACM/IEEE conference on Supercomputing* (November 1991, Albuquerque, New Mexico, USA), 1991, ISBN 978-0-89791-459-8, pp. 4–13. ↑45

[238] С. В. Полуян. *Анализ обращений программы к памяти в оптимизирующей распараллеливающей системе*, Диссертация на соискание степени кандидата технических наук по специальности 05.13.11: математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей, СПбГУ ИТМО, Санкт-Петербург, 2011, 147 с. ↑45,46

[239] С. В. Полуян. «Анализ обращений программы к памяти в оптимизирующей распараллеливающей системе», *Труды XVIII Всероссийской научно-методической конференции Телематика’2011.* Т. 2 (20–23 июня 2011, Санкт-Петербург, Россия), СПбГУ ИТМО, СПб., 2011, с. 328–329. ↑46

[240] С. В. Полуян. «Профилирование и его применение в диалоговом высокоуровневом оптимизирующем распараллеливателе», *Труды Международной суперкомпьютерной конференции «Научный сервис в сети Интернет: суперкомпьютерные центры и задачи»* (20–25

сентября 2010, Новороссийск, Россия), Изд-во МГУ, М., 2010, с. 652–653. ↑46

[241] B. Steinberg, A. Baglij, V. Petrenko, V. Burhovetckij, O. Steinberg, E. Metelica. “An analyzer for program parallelization and optimization”, *ICAIT’2018: Proceedings of the 3rd International Conference on Applications in Information Technology* (November 2018, Aizu-Wakamatsu, Japan), ISBN 978-1-4503-6516-1, pp. 90–95. ↑47,60

[242] R. Arora, J. Olaya, M. Gupta. “A tool for interactive parallelization”, *XSEDE’14: Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment* (July 2014, Atlanta, GA, USA), ISBN 978-1-4503-2893-7, pp. 1–8. ↑47

[243] L. R. Gervich, S. A. Guda, D. V. Dubrov, R. A. Ibragimov, E. A. Metelitsa, Yu. M. Mikhailuts, A. E. Paterikin, V. V. Petrenko, I. R. Skapenko, B. Ya. Steinberg, O. B. Steinberg, V. A. Yakovlev, M. V. Yurushkin. “How OPS (optimizing parallelizing system) may be useful for clang”, *CEE-SECR’17: Proceedings of the 13th Central & Eastern European Software Engineering* (October 2017, St. Petersburg, Russia), 2017, ISBN 978-1-4503-6396-9, pp. 1–8. ↑47

[244] З. Я. Нис. «Обеспечение статической семантической корректности программ при проведении распараллеливающих и оптимизирующих преобразований», *Труды четвертой международной конференции «Параллельные вычисления и задачи управления»*, PACO’2008 (27–29 октября 2008, Москва, Россия). ↑47

[245] А. Б. Бугеря. «Диалоговая отладка параллельных программ. Распределенная схема взаимодействующих компонентов», *Программирование*, **34**:3 (2008), с. 42–49 . ✻% ↑48

[246] M. Wolfe. “More iteration space tiling”, *Supercomputing’89: Proceedings of the 1989 ACM/IEEE conference on Supercomputing* (November 1989, Reno, Nevada, USA), 1989, ISBN 978-0-89791-341-6, pp. 665–664. ↑48

[247] J. Xue, Q. Huang, M. Guo. “Enabling loop fusion and tiling for cache performance by fixing fusion-preventing data dependences”, International Conference on Parallel Processing (ICPP’05) (14–17 June 2005, Oslo, Norway), 2005, pp. 107–115. ↑49

[248] K. Barton, J. Doerfert, H. Finkel, M. Kruse. *Loop Fusion, Loop Distribution and Their Place in the Loop Optimization Pipeline* (Accessed 1.07.2020), 20 pp. ↑49

[249] Б. Я. Штейнберг, О. Б. Штейнберг, А. А. Василенко. «Слияние циклов для локализации данных», *Программные системы: теория и приложения*, **11**:3 (46) (2020), с. 19–34. ↑49

[250] Л. К. Эйсымонт, В. С. Горбунов. «На пути к экзафлопсному суперкомпьютеру: результаты, направления, тенденции», МСКФ 2012 (Москва, 1 ноября 2012). ↑49

[251] A. Mycroft. “Programming language design and analysis motivated by hardware evolution”, *SAS 2007: Static Analysis*, International Static Analysis Symposium, Lecture Notes in Computer Science, vol. **4634**, Springer, Berlin–Heidelberg, 2007, ISBN 978-3-540-74061-2, pp. 18–33. ↑49

[252] А. И. Галушкин. «Стратегия развития современных супернейрокомпьютеров на пути к экзафлопным вычислениям», Приложение к журналу «Информационные технологии», 2012, 132 с. ↑49

[253] Н. Д. Морунов, Д. Л. Головашкин. «Особенности построения блочных алгоритмов FDTD-метода при организации вычислений на графическом процессоре с использованием языка MATLAB», *Компьютерная оптика*, **43**:4 (2019), с. 671–676. ↑49

[254] D. Kim, L. Renganarayanan, D. Rostron, S. Rajopadhye, M. M. Strout. “Multi-level tiling: M for the price of one”, *SC'07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing* (November 2007, Reno, Nevada, USA), 2007, ISBN 978-1-59593-764-3, pp. 1–12. ↑50

[255] М. В. Юрушкин. «Двойное блочное размещение данных в оперативной памяти при решении задачи умножения матриц», *Программная инженерия*, **7**:3 (2016), с. 132–139. ↑50

[256] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, P. Wu, I. Yamazaki, A. Yarkhan, M. Abalenkovs, N. Bagherpour, S. Hammarling, J. Šístek, D. Stevens, M. Zounon, and S. D. Relton. “PLASMA: Parallel linear algebra software for multicore using OpenMP”, *ACM Transactions on Mathematical Software (TOMS)*, **45**:2 (2019), pp. 1–35, 16. ↑

[257] М. В. Юрушкин. «Автоматизация блочного размещения данных в памяти компилятором языка Си», *Программная инженерия*, 2014, №6, с. 16–18. ↑50,52

[258] Л. Р. Гервич, Б. Я. Штейнберг, М. В. Юрушкин. *Разработка параллельных программ с оптимизацией использования структуры памяти*, Изд-во Южного федерального университета, Ростов-на-Дону, 2014, ISBN 978-5-9275-1500-4, 120 с. ↑50,54

[259] Б. Я. Штейнберг. *Оптимизация размещения данных в параллельной памяти*, Изд-во Южного федерального университета, Ростов-на-Дону, 2010, ISBN 978-5-9275-0687-3, 255 с. ↑50,54,56

[260] В. В. Пакулев, В. В. Воеводин. *Определение дуг графа алгоритма*, Препр. АН СССР, Отд. вычисл. математики, № 228, Отдел вычислительной математики АН СССР, М., 1989, 22 с. ↑51

[261] В. В. Воеводин. *Математические основы параллельных вычислений*, Изд-во МГУ, М., 1991, 345 с. ↑51

[262] P. Feautrier. “Some efficient solutions to the affine scheduling problem. I. One-dimensional time”, *International Journal of Parallel Programming*, **21**:5 (1992), pp. 313–347. ↑51

[263] P. Feautrier. “Some efficient solutions to the affine scheduling problem. II.

Multidimentional time", *International Journal of Parallel Programming*, **21**:6 (1992), pp. 389–420. ↑51

[264] P. Feautrier. "Parametric integer programming", *RAIRO-Operations Research*, **22**:3 (1988), pp. 243–268. ↑51

[265] Н. А. Лиходед. «Распределение операций и массивов данных между процессорами», *Программирование*, **29**:3 (2003), с. 73–80 . ↑51

[266] С. А. Гуда. «Оценки длины критического пути в решетчатом графе», *Труды четвертой международной конференции «Параллельные вычисления и задачи управления»*, PACO'2008 (Москва, 27–29 октября 2008), с. 1253–1267. ↑51,57

[267] Б. Я. Штейнберг. «О взаимосвязи между решетчатым графом программы и графом информационных связей», *Известия ВУЗов. Северокавказский регион. Естественные науки*, 2011, №5, с. 29–31. ↑51

[268] A. Fernandez, J. M. Llaberia, M. Valero-Garcia. "Loop transformation using nonunimodular matrices", *IEEE Transactions on Parallel and Distributed Systems*, **6**:8 (1995), pp. 832–840. ↑51

[269] A. Hartono, Muthu Manikandan Baskaran, J. Ramanujam, P. Sadayappan. "DynTile: Parametric tiled loop generation for parallel execution on multicore processors", *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)* (19–23 April 2010, Atlanta, GA, USA), 2010. ↑52

[270] U. Bondhugula, A. Hartono, J. Ramanujam, P. Sadayappan. "A practical automatic polyhedral parallelizer and locality optimizer", *ACM SIGPLAN Notices*, 2008, pp. 101–113. ↑52

[271] A. Acharya, U. Bondhugula, A. Cohen. "Polyhedral auto-transformation with no integer linear programming", *Proceedings of 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI'18, ACM, New York, NY, USA, 2018, pp. 529–542. ↑52

[272] V. Bandishti, I. Pananilath, U. Bondhugula. "Tiling stencil computations to maximize parallelism", *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Salt Lake City, Utah, USA, November 10–16, 2012). ↑52

[273] U. Bondhugula, A. Acharya, A. Cohen. "The Pluto+ algorithm: A practical approach for parallelization and locality optimization of affine loop nests", *ACM Trans. Program. Lang. Syst.*, **38**:3 (2016), 12, 32 pp. ↑52

[274] M. Wolfe, U. Banerjee. "Data dependence and its application to parallel processing", *International Journal of Parallel Programming*, **16**:2 (1987), pp. 137–178. ↑52

[275] С. Б. Арыков, В. Э. Малышкин. «Система асинхронного параллельного программирования «Аспект»», *Вычислительные методы и программирование*, **9**:1 (2008), с. 48–52. ↑52

[276] Л. Р. Гервич, Б. Я. Штейнберг, М. В. Юрушкин. «Программирование экзафлопсных систем», *Открытые системы. СУБД*, 2013, №8, с. 26–29. ↑52,54

[277] А. А. Корж. «Результаты масштабирования бенчмарка NPB UA на тысячи ядер суперкомпьютера Blue Gene/P с помощью расширения OpenMP», *Вычислительные методы и программирование*, **11** (2010), с. 31–41. ↑52

[278] V. Agarwal, F. Petrini, D. Pasetto, D. Bader. “Scalable graph exploration on multicore processors”, *SC’10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis* (13–19 Nov. 2010, New Orleans, LA, USA), 11 pp. ↑52

[279] Б. Я. Штейнберг. «Блочно-рекуррентные размещения матрицы для алгоритма Флойда», *Известия ВУЗов. Северо-Кавказский регион. Естественные науки*, 2010, №5, с. 31–33. ↑52,54

[280] A. P. Bagliy, A. V.Boukhanovsky, B. Ya. Steinberg, R. B. Steinberg. «Baltic sea water dynamics model acceleration», *Вестн. ЮУрГУ. Сер. Матем. моделирование и программирование*, **10**:1 (2017), с. 113–124. ↑52

[281] S. G. Ammaev, L. R. Gervich, B. Y. Steinberg. “Combining parallelization with overlaps and optimization of cache memory usage”, *PaCT 2017: Parallel Computing Technologies*, Lecture Notes in Computer Science, vol. **10421**, pp. 257–264. ↑52,54,60

[282] A. Yu. Perepelkina, V. D. Levchenko. «DiamondTorre algorithm for high-performance wave modeling», *Препринты ИПМ им. М. В. Келдыша*, 2015, 018, 20 с. ↑52

[283] Б. А. Корнеев, В. Д. Левченко. «Эффективное решение трехмерных задач газовой динамики Рунге-Кутты разрывным методом Галеркина», *Журнал вычислительной математики и математической физики*, **56**:3 (2016), с. 465–475 . % ↑52

[284] A. V. Zakirov, V. D. Levchenko, A. Yu. Perepelkina, Yasunari Zempo. «High performance FDTD code implementation for GPGPU supercomputers», *Препринты ИПМ им. М. В. Келдыша*, 2016, 044, 22 с. ↑52

[285] A. Yu. Perepelkina, V. D. Levchenko. «The DiamondCandy algorithm for maximum performance vectorized cross-stencil computation», *Препринты ИПМ им. М. В. Келдыша*, 2018, 225, 23 с. ↑52

[286] I. J. Bertolacci, C. Olschanowsky, B. Harshbarger, B. L. Chamberlain, D. G. Wonnacott, M. M. Strout. “Parameterized Diamond Tiling for Stencil Computations with Chapel parallel iterators”, *ICS’15: Proceedings of the 29th ACM on International Conference on Supercomputing* (June 8-11, 2015, Newport Beach, CA, USA), pp. 197–206. ↑52

[287] Ки-Чанг Ким. «Мелкозернистое распараллеливание неполных гнезд циклов», *Программирование*, 1997, №2, с. 52–66. ↑52

[288] S. L. Graham, M. Snir, C. A. Patterson. *Getting Up To Speed: The Future Of Supercomputing*, National Academies Press, 2005, ISBN 978-0309095020, 306 pp. ↑53

[289] Ш. Пипер, Дж. Пол, М. Сколт. «Новая эра в оценке производительности компьютерных систем», *Открытые системы. СУБД*, 2007, №9, с. 52–58. URL ↑53

[290] В. В. Корнеев. «Проблемы программирования суперкомпьютеров на базе многоядерных мультитредовых кристаллов», *Труды Всероссийской суперкомпьютерной конференции «Научный сервис в сети Интернет: масштабируемость, параллельность, эффективность»* (Новороссийск, 21–26 сентября 2009), Изд-во МГУ, М., 2009, с. 12–15. ✻ ↑54

[291] Б. Я. Штейнберг. «Блочно-аффинные размещения данных в параллельной памяти», *Информационные технологии*, 2010, №6, с. 36–41. URL ↑54,56

[292] В. А. Савельев. «Об оптимизации распараллеливания вычислений типа прямого метода в задаче теплопроводности для систем с распределенной памятью», *Известия вузов. Северо-Кавказский регион. Естественные науки*, 2012, №4, с. 12–14. ✻ ↑54

[293] U. Bondhugula. *Automatic distributed-memory parallelization and codegeneration using the polyhedral framework*, Technical report, ISc-CSA-TR-2011-3, 2011, 10 pp. URL ↑55

[294] T. Yuki, S. Rajopadhye. *Canonic multi-projection: memory allocation for distributed memory parallelization*, Technical Report CS11-106, Colorado State University, 12 pp. URL ↑55

[295] A. Syschikov, D. Rutkov. “Data allocation for parallel processing in distributed computing systems”, 6-th Seminar Of Finnish-Russian University Cooperation in Telecommunication, 2009, 12 pp. URL ↑55

[296] С. М. Зотов, И. С. Голосов, Ю. Б. Лифшиц, В. Д. Емельянов, С. Н. Орлов, А. Б. Мнев. «Распределение данных в автоматически параллелелизуемых программах для компьютеров с распределенной памятью», *Информационные технологии и информационные системы*, 1999, №1. ↑56

[297] P. Yunheung. *Compiling for Distributed Memory Multiprocessors Based on Access Region Analysis*, Thesis Ph.D in Computer Science, University of Illinois at Urbana-Champaign, 1997. ↑56

[298] Е. А. Метлицкий, В. В. Каверзнев. *Системы параллельной памяти: теория, проектирование, применение*, ЛГУ, Л., 1989, ISBN 9785288002472, 238 с. ↑56

[299] А. В. Фролов. «Оптимизация размещения массивов в ФОРТРАН-программах на многопроцессорных вычислительных системах», *Программирование*, 1998, №3, с. 70–80. ↑56

[300] U. Banerjee, S. C. Chen, D. J. Kuck, R. A. Towle. “Time and parallel

processor bounds for Fortran-like loops", *IEEE Transactions on Computers*, **C-28**:9 (1979), pp. 660–670. ↑57

[301] А. А. Самарский. *Введение в численные методы*, Наука, М., 1987, ISBN 5-8114-0602-9, 280 с. ↑57

[302] Н. М. Ершов, А. М. Попов. «Оптимизация параллельных вычислений с учетом архитектуры и быстродействия связей в вычислительной системе», *Вестник Московского университета: Вычислительная математика и кибернетика*, 1993, №1, с. 24–30. ↑58

[303] И. И. Левин, Б. Я. Штейнберг. «Сравнительный анализ эффективности параллельных программ для различных архитектур параллельных ЭВМ», *Искусственный интеллект*, 2001, №3, с. 234–242. ↑58

[304] Б. Я. Штейнберг. «Распараллеливание рекуррентных циклов с условными операторами», *Автоматика и телемеханика*, 1995, №6, с. 176–184. ↑58

[305] V. Strassen. "Gaussian elimination is not optimal", *Numer. Math.*, **13**:4 (1969), pp. 354–356. ↑58

[306] А. Ахо, Дж. Хопкрофт, Дж. Ульман. *Построение и анализ вычислительных алгоритмов*, Мир, М., 1979, ISBN 978-5-458-26604-8, 536 с. ↑58

[307] Л. А. Разборов. «P?=NP или проблема: взгляд из 90-х», *Математические события 20-ого Века*, ФАЗИС, М., 2003, с. 399–416. ↑58

[308] Б. Я. Штейнберг. «Зависимость оптимального распределения площади кристалла процессора между памятью и вычислительными ядрами от алгоритма», *Труды международной конференции «Параллельные вычисления и задачи управления»*, PACO'2012 (24–26 октября 2012, ИПУ РАН, Москва), с. 99–108. ↑58

[309] В. М. Чернов. «Параллельная машинная арифметика для рекуррентных систем счисления в неквадратичных полях», *Компьютерная оптика*, **44**:2 (2020), с. 274–281. ↑59

[310] V. V. Burkhovetskiy, B. Ya. Steinberg. "Parallelizing an exact algorithm for the traveling salesman problem", 6-th International Young Scientist Conference on Computational Science YSC 2017 (01–03 November 2017, Kotka, Finland), *Procedia Computer Science*, **119** (2017), pp. 97–102. ↑60

[311] V. V. Burkhovetskiy, B. Ya. Steinberg. "An exact parallel algorithm for traveling salesman problem", *CEE-SECR'17: Proceedings of the 13th Central & Eastern European Software Engineering* (20–22 Oct. 2017, St. Peterburg, Russian Federation), 5 pp. ↑60

[312] В. Бурховецкий, Б. Штейнберг. «Стратегия использования крупных заданий при параллельном обходе дерева», *Труды Всероссийской научной конференции памяти А. Л. Фуксмана «Языки программирования и компиляторы»* (3–5 апреля 2017, Южный федеральный университет,

Ростов-на-Дону), Издательство Южного федерального университета, Ростов-на-Дону, 2017, с. 66–70. ↑60

[313] B. J. Steinberg, J. M. Abu-Khalil, M. G. Adigeyev, A. A. Bout, A. V. Kermanov, E. A. Pshenichnyy, G. V. Ramanchauskayte, A. P. Kroshkina, A. V. Gutnikov, N. S. Ponomareva, A. E. Panich, T. Shkurat. "A package of fast tools for genomic sequence analysis", *Int. J. Math. Models Methods Appl. Sci.*, **10** (2016), pp. 42–50. ↑61

[314] M. Farrar. "Striped Smith-Waterman speeds database searches six times over other SIMD implementations", *Bioinformatics*, **23**:2 (2007), pp. 156–61. ↑61

[315] J. Daily. "Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments", *BMC Bioinformatics*, **17** (2016), 81, 11 pp. ↑61

[316] Y. Liu, B. Schmidt, D. L. Maskell. "CUDASW++ 2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions", *BMC Research Notes*, **3**:1 (2010), 93. ↑61

[317] H. Lan, W. Liu, Y. Liu, B. Schmidt. "SWhybrid: a hybrid-parallel framework for large-scale protein sequence database search", 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (29 May–2 June 2017, Orlando, FL, USA), pp. 42–51. ↑61

Поступила в редакцию	19.09.2020
Переработана	22.12.2020
Опубликована	02.03.2021

Рекомендовал к публикации *к.ф.-м.н. С. А. Романенко*

Пример ссылки на эту публикацию:

Б. Я. Штейнберг, О. Б. Штейнберг. «Преобразования программ — фундаментальная основа создания оптимизирующих распараллеливающих компиляторов». *Программные системы: теория и приложения*, 2021, **12**:1(48), с. 21–113. 10.25209/2079-3316-2021-12-1-21-113
http://psta.psiras.ru/read/psta2021_1_21-113.pdf

Об авторах:

Борис Яковлевич Штейнберг

д.т.н, зав. каф., с.н.с. Института математики, механики и компьютерных наук Южного федерального университета. Научные интересы: распараллеливание программ, дискретная оптимизация.

0000-0001-8146-0479
e-mail: borsteinb@mail.ru

Олег Борисович Штейнберг

к.ф.-м.н., доцент ИММиКН ЮФУ. Научные интересы: преобразования программ, алгоритмы на графах.

0000-0003-4152-0988
e-mail: olegsteinb@gmail.com

CSCSTI 50.07.05
UDC 519.681.3+004.415.3

Boris Ya. Steinberg, Oleg B. Steinberg. *Program transformations as the base for optimizing parallelizing compilers.*

Abstract. The paper deals with program transformations leading to acceleration and summarize the publications on various parallel computing architectures and tools for developing effective programs for them. The discussion focuses on a combination of parallelization and optimization of access to memory modules of different levels. It highlights that the lag of automatic program optimization from the needs of new architectures restrains the development of new promising computing systems.

The development of the theory of program transformation and optimizing (parallelizing) compilers could lead to a significant increase in the productivity of programmers. The article substantiates the call for the modernization of the optimizing compilation and presents new problem statements.

Key words and phrases: optimizing compiler, parallel computations, program transformations, data locality, memory access optimization, tile.

2020 *Mathematics Subject Classification*: 97P30; 97P20, 97R40

References

[1] *Big tasks and supercomputers*, Per. s angl., Trudy instituta inzhenerov po elektrotekhnike i radioelektronike, vol. **77**, no. 7, Mir, M., 1989. ↑23

[2] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979, ISBN 978-5-458-26100-5 ISBN 0-7167-1045-5, x+338 pp. ↑23,42

[3] Yu. Zatuliveter, Ye. Fishchenko. "Multiprocessor computer PS-2000", *Open Systems.DBMS*, 2007, no. 9, pp. 74–79 (in Russian). ↑24,57

[4] H. J. Siegel. "A model of SIMD machines and a comparison of various interconnection networks", *IEEE Transactions on Computers*, **C-28**:12 (1979), pp. 907–917. ↑24

[5] I. V. Prangishvili, S. Ya. Vilenkin, I. L. Medvedev. *Parallel computing systems with common control*, Energoatomizdat, M., 1983 (in Russian), 312 pp. ↑25,26,27

[6] A. V. Zabrodin, V. V. Karatanov, V. V. Korneyev, V. K. Levin. "Massively parallel computing systems based on serial microprocessors. Experience in creation and application", *Trudy Mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2001 (2–4 oktyabrya 2001, Moskva, IPU RAN), pp. 85–86 (in Russian). ↑25

[7] A. V. Zabrodin, V. K. Levin. "Experience in the development of parallel computing technologies. Creation and development of the MBS family", *Trudy Vserossiyskoy nauchnoy konferentsii «Vysokoproizvoditel'nyye vychisleniya i ikh prilozheniya"* (30 oktyabrya–2 noyabrya 2000 goda, Chernogolovka), pp. 3–8 (in Russian). ↑25

[8] A. O. Latsis. *How to build and use a supercomputer*, Best·seller, M., 2003, ISBN 5-98158-003-8 (in Russian), 240 pp. ↑25

[9] M. A. Kopytov, G. M. Mikhaylov, Yu. P. Rogov. "High-performance cluster of the A. A. Dorodnitsin Computing Center of the Russian Academy of Sciences", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 22–27 sentyabrya 2003), MGU, M., pp. 60–62 (in Russian). ↑25

[10] S. M. Abramov, A. I. Adamovich, M. R. Kovalenko, A. F. Slepukhin, N. N. Paramonov. "Cluster systems of the SKIF family of supercomputers", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 22–27 sentyabrya 2003), MGU, M., pp. 147–151 (in Russian). ↑25

[11] A. A. Bukatov, V. N. Datsyuk, O. V. Datsyuk, G. M. Khachkinayev. "Experience in creating a high-performance cluster using two communication networks", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 22–27 sentyabrya 2003), MGU, M., pp. 110–112. ↑25

[12] M. P. Filamofit·skiy. "X-COM System: Organization of Distributed Computing on the Internet", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 22–27 sentyabrya 2003), MGU, M., pp. 363–367. ↑25

[13] Yu. S. Zatuliveter. "To the global computer", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 22–27 sentyabrya 2003), MGU, M., pp. 186–190. ↑25

[14] Yu. N. Galyuk, V. I. Zolotarev, V. P. Menonov. "GRID: Resilient Multi-Cluster Computing Scheme", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 22–27 sentyabrya 2003), MGU, M., pp. 87. ↑25

[15] R. Allen, K. Kennedy. *Optimizing Compilers for Modern Architectures: A Dependence-based Approach*, 1st ed., Morgan Kaufmann Publisher, 2001, ISBN 978-1558602861, 816 pp. ↑25,40,42,45

[16] P. M. Kougi. *The architecture of conveyor computers*, Per. s angl., Radio i svyaz', M., 1985, 358 pp. ↑25,27

[17] K. G. Samofalov, G. M. Lutskiy. *Fundamentals of the theory of multilevel pipelined computing systems*, Radio i svyaz', M., 1989, 272 pp. ↑25

[18] I. A. Kalyayev, I. I. Levin, Ye. A. Semernikov, V. I. Shmoylov. *Reconfigurable multicore computational structures*, 2-ye, pererab. i dop. izd., ed. I. A. Kalyayev, Izd-vo YuNTs RAN, Rostov-n/D, 2009, 344 pp. ↑25,37

[19] B. A. Babayan. "The level of programming and architectural principles of building a computer", Kibernetika i vychislitel'naya tekhnika, vol. **2**, Nauka, M., 1986, pp. 18–27. ↑26

[20] B. A. Babayan. "Main Principles of E2K Architecture", *Free Software Magazine*, **1**:2 (2002). URL ↑26,27

[21] B. Babayan, A. V. Bocharov, V. S. Volin, S. S. Gavrilov, A. S. Groshev, F. A. Gruzdov, M. V. Yeremin, S. M. Zotov, A. L. Plotkin, L. Ye. Pshenichnikov, G. G. Ryabov, M. L. Chudakov, V. S. Shevyakov, *Multiprocessor computers and methods for their design*, Perspektivy razvitiya vychislitel'noy tekhniki, vol. **4**, Vysshaya shkola, M., 1990, ISBN 5-06-000133-4, 144 pp. ↑26

[22] P. C. Treleaven. "Parallel architecture overview", *Parallel Computing*, **8**:1988, pp. 59–70. ↑26,27

[23] V. S. Podlazov. "Properties of multi-ring and hypercube commutators on arbitrary permutations", *Trudy Mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2001 (2–4 oktyabrya 2001, Moskva, IPU RAN), pp. 152–164. ↑26

[24] A. A. Arkhangel'skaya, V. A. Yershov, V. I. Neyman. *Automatic switching of communication channels*, Svyaz', M., 1970, 192 pp. ↑26

[25] V. E. Benesh. *Mathematical foundations of the theory of telephone communications*, Svyaz', M., 1968, 292 pp. ↑26

[26] V. I. Kodachigov. *Electronic switching of information channels*, Izd-vo Rostovskogo universiteta, Rostov-na-Donu, 1983, 208 pp. ↑26

[27] A. V. Kalyayev. *Homogeneous Switching Register Structures*, Sovet·skoye radio, M., 1978, 334 pp. ↑26,37

[28] G. T. Artamonov. *Topology of regular computer networks and environments*, Radio i svyaz', M., 1985, 192 pp. ↑26

[29] G. T. Artamonov, V. D. Tyurin. *Topology of computer networks and multiprocessor systems*, Radio i svyaz', M., 1991, 248 pp. ↑26

[30] V. V. Korneyev, V. G. Khoroshevskiy. *Architecture of computing systems with programmable structure*, preprint OVS-10, Institut matematiki SO AN SSSR, Novosibirsk, 1979, 48 pp. ↑26,37

[31] V. V. Korneyev, V. G. Khoroshevskiy. *Structure and functional organization of computer systems with programmable structure*, preprint OVS-11, Institut matematiki SO AN SSSR, Novosibirsk, 1979, 48 pp. ↑26,37

[32] Sh. Akhter, R. Jason. *Multi-Core Programming. Increasing Performance Through Software Multi-threading*, Intel-Press, USA, 2006, 336 pp. ↑26

[33] V. Yu. Volkonskiy, A. K. Kim. "Development of ideas of parallelism in the architecture of computing complexes of the "Elbrus" series", *Trudy chetvertoy mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2008 (Moskva, 27–29 oktyabrya 2008), pp. 42–66. ↑27

[34] S. A. Orlov, B. Y Tsilker. *Organization of computers and systems: Textbook for universities*, fundamental course on the architecture and structure of modern computer facilities, 3-rd, Piter, SPb., 2015, ISBN 978-5-496-01145-7 (in Russian), 685 pp. ↑27

[35] A. I. Solonina, D. A. Ulakhovich, L. A. Yakovlev. *Algorithms and processors for digital signal processing*, BKhV- Peterburg, SPb., 2002, ISBN 5-94157-065-1, 454 pp. ↑27

[36] V. V. Voyevodin. *Mathematical models and methods in parallel processes*, Nauka, Glavnaya redaktsiya fiziko-matematicheskoy literatury, M., 1986, 296 pp. ↑27,31,51,58

[37] B. A. Golovkin. *Parallel computing systems*, Nauka, Glavnaya redaktsiya fiziko-matematicheskoy literatury, M., 1980, 518 pp. ↑27

[38] V. V. Korneyev. *Parallel computing systems*, Nolidzh, M., 1999, ISBN 5-89251-065-4, 320 pp. ↑27

[39] D. Ivens (red.). *Parallel processing systems*, Mir, M., 1985, 413 pp. ↑27

[40] R. W. Hockney, C. R. Jesshope. *Parallel computers: Arhitecture, programming and algorithms*, Adam Hilger, 1981, ISBN 978-0852744222, 423 pp. ↑27

[41] G. S. Almasi, A. Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Series in Computer Science and Engineering, Subsequent edition, Benjamin-Cummings Pub Co, 1993, ISBN 978-0805304435, 689 pp. ↑27,39

[42] Y. Wallach, *Alternating Sequential-Parallel Processing*, Lecture Notes in Computer Science, vol. **127**, ISBN 978-3-540-38988-0, xii+336 pp. ↑27

[43] Kun Sun' Yuan'. *Matrix processors on VLSI*, Mir, M., 1991, ISBN 5-03-001857-3, 672 pp. ↑27

[44] V. N. Beletskiy. *Multiprocessor and parallel structures with the organization of asynchronous computations*, Naukova dumka, Kiyev, 1988, ISBN 5-12-009328-0, 240 pp. ↑27

[45] N. N. Mirenkov. *Parallel programming for multi-module computing systems*, Radio i svyaz', M., 1989, ISBN 5-256-00196-5, 320 pp. ↑27

[46] J. Mukundan, H. Hunter, K.-H. Kim, J. Stuecheli, J. F. Martínez. "Understanding and Mitigating Refresh Overheads in High-Density DDR4 DRAM Systems", *ACM SIGARCH Computer Architecture News*, **43**:3 (2013). ↑27

[47] A. V. Linev, D. K. Bogolyubov, S. I. Bastrakov, *Parallel programming technologies for new processor architectures*, Uchebnik, Superkomp'yuternoye obrazovaniye, ed. V. P. Gergel', Izd-vo Moskovskogo un-ta, M., 2010, ISBN 978-5-211-05962-7, 152 pp. ↑28

[48] A. Olofsson. *Epiphany-V: A 1024-core 64-bit RISC processor*, 2016, 15 pp. arXiv 1610.01832 ↑28,56

[49] A. A. Adamov, P. V. Pavluxin, D. V. Bikonov, A. L. Eisymont, L. K. Eisymont. "Modern GPGPU alternative perspective universal and specialized processors-accelerators", *Voprosy kiberbezopasnosti*, 2019, no. 4, pp. 13–21. ↑28,56

[50] M. Kuzminskij. "Power9 — processors for big data", *Otkrytye systemy. SUBD*, 2017, no. 03 (Accessed 1.07.2020). https://www.osp.ru/os/2017/03/13052698 ↑28

[51] I. Grasso, P. Radojkovic, N. Rajovic, I. Gelado, A. Ramírez. "Energy efficient HPC on embedded SoCs: Optimization techniques for Mali GPU", *2014 IEEE 28th International Parallel and Distributed Processing Symposium*. ↑28

[52] D. Kharris, S. Kharris. *Digital circuitry and computer architecture*, DMK Press, 2018, ISBN 978-5-97060-570-7, 792 pp. ↑28

[53] S. S. Andreyev, S. A. Dbar, Yu. A. Klimov, A. O. Latsis, Ye. A. Plotkina. "Quantum computation model: the classical programmer's viewpoint", *Keldysh Institute preprints*, 2018, 178 (in Russian), 30 pp. ↑28

[54] S. A. Stepanenko. "Photonic computer: structure and algorithms. Estimations of parameters", *Fotonika*, 2017, no. 7, pp. 72–83 (in Russian). ↑28,38

[55] Ye. Zuyev. *Rare profession*, DMK-Press, 2014, ISBN 978-5-94074-812-0. ↑29

[56] V. A. Val'kovskiy. *Parallelization of Algorithms and Programs: A Structural Approach*, Radio i svyaz', M., 1989, ISBN 978-5-256-00195-7, 176 pp. ↑29,36,54

[57] M. R. Haghighat, C. D. Polychronopoulos. "Symbolic analysis for parallelizing compilers", *ACM Transactions on Programming Languages and Systems*, **18**:4 (1995), pp. 477–518. ↑29,49

[58] E. A. Trakhtengerts. *Parallel process software*, Nauka, M., 1987. ↑29

[59] A. V. Babichev, V. G. Lebedev. "Parallelization of program loops", *Programmirovaniye*, **9**:5 (1983), pp. 52–63. ↑29

[60] S. Abramov, A. Adamovich, M. Kovalenko. “T-system: programming environment providing automatic dynamic parallelizing on IP-network of Unix-computers”, 4-aya Mezhdunarodnaya Rossiysko-Indiyskaya vystavka-seminar «Matematicheskoye modelirovaniye i vizualizatsiya” (15–25 sentyabrya 1997 g., Moskva). ↑31

[61] M. Ye. Balashov, V. D. Goryachev, D. S. Rykov, O. S. Rykova, Ye. M. Smirnov, N. Ye. Smirnova, S. A. Yakubov. “Preprocessor for solver CCFDD network IVS”, *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet”* (Novorossiysk, 22–27 sentyabrya 2003), Izd-vo MGU, M., pp. 62–64. ↑31

[62] S. L. Golovkov, A. G. Smirnov V. K. Rubin. “Monitor support for parallel scientific and technical problems on the Internet”, *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet”* (Novorossiysk, 22–27 sentyabrya 2003), Izd-vo MGU, M., pp. 79–82. ↑31

[63] A. S. Igumnov. “Open platform for debugging parallel programs”, *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet”* (Novorossiysk, 22–27 sentyabrya 2003), Izd-vo MGU, M., pp. 92–94. ↑31

[64] V. P. Gergel’, A. N. Svistunov. “Development of an integrated high-performance computing environment for a cluster of Nizhny Novgorod University”, Mezhdunarodnyy nauchno-prakticheskiy seminar i molodezhnaya shkola «Vysokoproizvoditel’nyye parallel’nyye vychisleniya na klasternykh sistemakh” (22–26 noyabrya 2005, NNGU im. N.I. Lobachevskogo), pp. 51–54. ↑31

[65] A. I. Il’yushin, V. N. Klepikov, A. V. Vyrodov, A. S. Babanin, S. A. Budikhin, K. V. Tachennikov. “"Pumping" of objects as a means of increasing the throughput of the MPS”, *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet”* (Novorossiysk, 22–27 sentyabrya 2003, Nizhniy Novgorod), Izd-vo MGU, M., pp. 345–347. ↑31

[66] V. A. Yevstigneyev, I. A. Mirzuitova. *Loop analysis: a selection of candidates for paralization*, Preprint No 58, ISI RAN, Novosibirsk, 1999, 49 pp. ↑31,36

[67] V. A. Yevstigneyev. “Some features of computer software with a long command word (Review)”, *Programmirovaniye*, **17**:2 (1991), pp. 69–80. ↑31

[68] A. W. Lim, G. I. Cheong, M. S. Lam. “An affine partitioning algorithm to maximize parallelism and minimize communication”, *ICS’99: Proceedings of the 13th international conference on Supercomputing* (Rhodes Greece, June, 1999), 1999, ISBN 978-1-58113-164-2, pp. 228–237. ↑31

[69] A. W. Lim, M. S. Lam. “Cache Optimizations With Affine Partitioning”, *Proceedings of the Tenth SIAM Conference on Parallel Processing for Scientific Computing, Portsmouth, Virginia, March, 2001.*, 2001, 14 pp. ↑31

[70] A. W. Lim, M. S. Lam. “Maximizing parallelism and minimizing synchronization with affine partitions”, *Parallel Computing*, **24**:3–4 (1998), pp. 445–475. ↑31

[71] A. V. Frolov. “Automation of FORTRAN program transformations”, *Tezisy dokladov vserossiyskoy nauchnoy konferentsii «Fundamental’nyye i prikladnyye aspekty razrabotki bol’shikh raspredelennykh programmnykh kompleksov”* (Novorossiysk, 21–26 sentyabrya 1998), MGU, M., 1998. ↑32,51

[72] A. V. Frolov. “Tool system for parallelizing Fortran programs as an example of using Internet technologies in programming”, *Tezisy dokladov vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet”* (Novorossiysk, 24–29 sentyabrya 2001), MGU, M., 2001, pp. 221–222. ↑32

[73] A. V. Frolov. “Finding and using directed cuts of real graphs of algorithms”, *Programmirovaniye*, **23**:4 (1997), pp. 71–80. ↑32

[74] V. V. Koval'. "Modern methods of program transformation", *Komp'yuternoye modelirovaniye. Vychislitel'nyye tekhnologii*, TsVVR, Rostov-na-Donu, 2003, pp. 41–58.↑32

[75] V. V. Lugovoy. "Methods for implementing the internal representation of programs in a CASE-system for parallelizing programs", *Komp'yuternoye modelirovaniye. Vychislitel'nyye tekhnologii*, 2003, pp. 91–99.↑32

[76] A. A. Bukatov, V. V. Koval'. "The methods of realization of transformation machine of multiple objective system of programs' transformations", *Informatsionnyye tekhnologii*, 2004, no. 3, pp. 24–30 (in Russian).↑32

[77] A. A. Bukatov, V. V. Lugovoy. "Incremental modification of data structures of the internal representation of programs in the program parallelization system", *Iskusstvennyy intellekt*, 2003, no. 3, pp. 15–22.↑32

[78] B. Ya. Shteynberg. "Vertices of the region of changing the parameters of cycles and information independence", *Tezisy dokladov I-y Vsesoyuznoy konferentsii «Odnorodnyye vychislitel'nyye sredy i sistolicheskiye struktury"*. V. 3 (17–20 aprelya 1990, g. L'vov), pp. 112–116.↑32

[79] B. Ya. Shteynberg. "Parallelization with analysis of the range of parameters of loops and with the analysis of external variables in index expressions", *Razrabotka sistemnogo i prikladnogo programmnogo obespecheniya MVK PS-2000/2100, PS-3000/3100*, Tezisy dokladov Vsesoyuznogo nauchno-tekhnicheskogo seminara (g. Kalinin), M., 1990, pp. 5–6.↑32

[80] B. Ya. Shteynberg. "Optimal parallel relocation of two-dimensional arrays", *Programmirovaniye*, **19**:6 (1993), pp. 81–87 (in Russian).↑32,56,57

[81] B. Ya. Shteynberg. "Conflict-free array allocations in parallel computations", *Kibernetika i sistemnyy analiz*, 1999, no. 1, pp. 166–178.↑32,56

[82] B. Ya. Shteynberg. "Optimal parallel relocation of multidimensional arrays in parallel computations", *Sbornik trudov Mezhdunarodnoy nauchno-tekhnicheskoy konferentsii «Intellektual'nyye mnogoprotsessornyye sistemy"* (1–5 sentyabrya, 1999, Taganrog, Rossiya), pp. 151–155.↑32,56,57

[83] B. Ya. Shteynberg. "Breaking loops for execution on a supercomputer with a reconfigurable pipeline architecture", *Iskusstvennyy intellekt*, 2002, no. 3, pp. 331–338.↑32

[84] B. Ya. Shteynberg. "Substitution and Renaming of Index Variables in Multidimensional Loops", *Izvestiya vuzov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2002, Yubileynyy vypusk, pp. 94–99.↑32,51

[85] B. Ya. Shteynberg. "Substitution and renaming in multidimensional loops with automatic parallelization", *Materialy Mezhdunarodnoy nauchno-tekhnicheskoy konferentsii «Super·YeVM i mnogoprotsessornyye vychislitel'nyye sistemy"*, MVS'2002 (Rossiya, Taganrog, 26–30 iyunya 2002), pp. 161–164.↑32

[86] B. Ya. Shteynberg. "The parallelizing of recurrent program loops", *Informatsionnyye tekhnologii*, 2004, no. 4, pp. 16–23 (in Russian).↑32,58

[87] B. Ya. Shteynberg. *Mathematical methods for parallelizing recurrent program loops for supercomputers with parallel memory*, Izd-vo Rostovskogo universiteta, Rostov-na-Donu, 2004, 192 pp.↑32,40,42,45,58

[88] B. Ya. Shteynberg, D. V. Makoshenko, D. N. Cherdantsev, A. M. Shul'zhenko. "Internal representation in an open parallelization system", *Iskusstvennyy intellekt*, 2003, no. 3, pp. 89–96.↑32

[89] B. Ya. Shteynberg, D. N. Cherdantsev, S. A. Naumenko, A. E. Butov, V. V. Petrenko. "Program conversions for an open parallelization system", *Iskusstvennyy intellekt*, 2003, no. 3, pp. 97–104. ↑32

[90] B. Ya. Shteynberg, M. V. Naprasnikova. "The minimum set of control arcs when testing software modules", *Izvestiya VUZov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2003, no. 4, pp. 15–18. ↑32,48

[91] B. Ya. Shteynberg, O. E. Arutyunyan, A. E. Butov, K. Yu. Gufan, R. Morylev, S. A. Naumenko, V. V. Petrenko, A. Tuzayev, D. N. Cherdantsev, M. V. Shilov, R. B. Shteynberg, A. M. Shul'zhenko. "OPC-based parallelization trainer", *Trudy nauchno-metodicheskoy konferentsii «Sovremennyye informatsionnyye tekhnologii v obrazovanii: Yuzhnyy federal'nyy okrug"* (Rostov-na-Donu, 12–15 maya 2004), pp. 248–250. ↑32,51

[92] B. Ya. Shteynberg, M. V. Naprasnikova, Z. Ya. Nis. "Testing transformations of the Open Parallelizing System", *Iskusstvennyy intellekt*, 2004, no. 3, pp. 257–264. ↑32,48

[93] B. Ya. Shteynberg. "Open parallelization system", *Otkrytyye sistemy. SUBD*, 2007, no. 9, pp. 36–41. URL ↑32,51

[94] B. Ya. Shteynberg. "Parallel multiplication of a sparse matrix by a vector", *Izvestiya vuzov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2005, Spetsial'nyy vypusk, pp. 122–124. ↑32

[95] B. Ya. Shteynberg, Z. Ya. Nis, V. V. Petrenko, D. N. Cherdantsev, R. B. Shteynberg, A. M. Shul'zhenko. "State and capabilities of the open parallelization system (summer 2006)", *Trudy seminara «Naukoyemkoye programmnoye obespecheniye" v ramkakh shestoy mezhdunarodnoy konferentsii pamyati akademika A. P. Yershova «Perspektivy sistem informatiki"* (Novosibirsk, Akademgorodok, 28–29 iyunya 2006), pp. 122–125. ↑32,51

[96] B. Ya. Shteynberg, Z. Ya. Nis, V. V. Petrenko, D. N. Cherdantsev, R. B. Shteynberg, A. M. Shul'zhenko. "Open parallelization system 2006", *Trudy III mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2006 (2–4 oktyabrya 2006, IPU RAN, Moskva), pp. 526–541. ↑32,51

[97] B. Ya. Shteynberg, R. I. Morylev. "Parallelization of programs using the Open Parallelizing System", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 18-22 sentyabrya 2007), Izd-vo MGU, M.. ↑32,47,51

[98] A. M. Shul'zhenko. "Automatic detection of ParDo cycles in the program", *Izvestiya vuzov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2005, no. S11, pp. 77–87. ↑32,51

[99] R. B. Shteynberg. "Calculation of the delay in the start of pipelines for supercomputers with a structurally procedural organization of computations", *Iskusstvennyy intellekt*, 2003, no. 4, pp. 105–112. ↑32,37

[100] O. B. Shteynberg. "Automatic parallelization of recurrent loops with irregular computation of superpositions", *Trudy chetvertoy mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2008 (Moskva, 27–29 oktyabrya 2008). ↑32,47,58

[101] A. N. Andrianov, K. N. Yefimkin, I. B. Zadykhaylo. "Non-procedural language for solving problems of mathematical physics", *Programmirovaniye*, **17**:2 (1991), pp. 80–94. ↑32

[102] A. N. Andrianov. "Using the Norma language for solving computational problems on irregular grids", *Trudy Vserossiyskoy nauchnoy konferentsii «Fundamental'nyye i*

prikladnyye aspekty razrabotki bol'shikh raspredelennykh programmnykh kompleksov" (Novorossiysk, 21–26 sentyabrya 1998), Izd-vo MGU, M., pp. 120–123. ↑32

[103] A. N. Andrianov, T. P. Baranova, A. B. Bugerya, K. N. Yefimkin. "Nonprocedural norma language and its translation methods for parallel architectures", *Izvestiya vysshikh uchebnykh zavedeniy. Severo-Kavkazskiy region. Tekhnicheskiye nauki*, 2017, no. 3 (195), pp. 5–12 (in Russian). ↑32

[104] V. F. Aleksakhin, V. N. Il'yakov, N. A. Konovalov, N. V. Kovaleva, V. A. Kryukov, N. V. Podderyugina, Yu. L. Sazanov. "Automation system for the development of parallel programs for computing clusters and networks (DVM-system)", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 23–28 sentyabrya 2002), Izd-vo MGU, M., 2002, pp. 272–273. ↑32,55

[105] V. N. Kovalenko I. V. Kryukov V. A. Il'yakov. "Analysis and prediction of the efficiency of DVM-programs execution on heterogeneous computer networks", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 22–27 sentyabrya 2003), Izd-vo MGU, M., 2003, pp. 181–182. ↑32,55

[106] V. A. Bakhtin, D. A. Zakharov, A. A. Yermichev, V. A. Kryukov. "Debugging parallel programs in DVM-system", *Elektronnyye biblioteki*, **23**:4 (2020), pp. 866–886 (in Russian). ↑32,33,55

[107] V. A. Bakhtin, D. A. Zakharov, A. S. Kolganov, V. A. Kryukov, N. V. Podderyugina, M. N. Pritula. "Development of parallel applications using DVM-system", *Vestn. YuUrGU. Ser. Vych. matem. inform.*, **8**:1 (2019), pp. 89–106 (in Russian). ↑32,33,55

[108] V. A. Bakhtin, D. A. Zakharov, A. A. Yermichev, V. A. Kryukov, N. V. Podderyugina, M. N. Pritula. "Experience in solving applied problems using irregular grids using a DVM system", Short articles and descriptions of posters of the XIII International Scientific Conference "Parallel Computing Technologies", PaVT'2018, 2018, pp. 241–252. ↑32,33,55

[109] V. A. Bakhtin, D. A. Zakharov, V. A. Kryukov, N. V. Podderyugina, M. N. Pritula. "Incremental parallelization of programs using DVM-system", *Trudy mezhdunarodnoy konferentsii «Superkomp'yuternyye dni v Rossii"*, Superkomp'yuternyy konsortsium universitetov Rossii, Rossiyskaya akademiya nauk, 2018, pp. 991–993. ↑32,33,55

[110] V. A. Bakhtin, O. F. Zhukova, N. A. Katayev, A. S. Kolganov, V. A. Kryukov, M. Yu. Kuznetsov, N. V. Podderyugina, M. N. Pritula, O. A. Savitskaya, A. A. Smirnov. "Parallelization of software packages. Problems and prospects", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (17-22 sentyabrya 2018 g., g. Novorossiysk), IPM im. M. V. Keldysha, M., 2018, ISBN 978-5-98354-046-0, pp. 63–72 (in Russian). ↑32,33,55

[111] A. A. Yermichev, V. A. Kryukov. "Development of the comparative debugging method for DVMH programs", *Trudy XIX Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (18–23 sentyabrya 2017 g., g. Novorossiysk), IPM im. M. V. Keldysha, M., 2017, ISBN 978-5-98354-037-8, pp. 150–156. ↑32,33,55

[112] A. S. Kolganov, V. A. Kryukov, K. O. Shokhin. "Parallelizing loops with regular data dependencies on clusters with GPUs", *Tezisy dokladov nauchnoy konferentsii «Lomonosovskiye chteniya"* (Moskva, 17–26 aprelya 2017 goda), OOO «MAKS Press", M., 2017, ISBN 978-5-89407-572-3, pp. 29–30. ↑32,33,55

[113] V. A. Bakhtin, O. F. Zhukova, A. S. Kolganov, N. N. Korolev, V. A. Kryukov, M. Yu. Kuznetsov, N. V. Podderyugina, M. N. Pritula, O. A. Savitskaya, A. A. Smirnov,

N. A. Katayev. “Incremental parallelization for clusters in SAPFOR”, *Trudy XIX Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet”* (18–23 sentyabrya 2017 g., g. Novorossiysk), IPM im. M. V. Keldysha, M., 2017, ISBN 978-5-98354-037-8, pp. 48–52. ↑32,33,55

[114] V. F. Aleksakhin, V. A. Bakhtin, D. A. Zakharov, A. S. Kolganov, A. V. Korolev, V. A. Kryukov, N. V. Podderyugina, M. N. Pritula. “HPC applications experience using DVM-system”, *Trudy mezhdunarodnoy konferentsii «Superkomp'yuternyye dni v Rossii”*, 2017, pp. 650–661 (in Russian). ↑32,33,55

[115] V. Bakhtin, A. Kolganov, V. Krukov, N. Podderyugina, M. Pritula, O. Savitskaya. “An extension of the DVM-system to solve problems with intensive irregular memory access”, GraphHPC 2017 Proceedings of the 4th GraphHPC Conference on Large-Scale Grap Processing Using HPC Systems, CEUR Workshop Proceedings, vol. **1981**, 2017, pp. 25–30. ↑32,33,55

[116] V. A. Bakhtin, A. S. Kolganov, V. A. Kryukov, N. V. Podderyugina, M. N. Pritula. “Extension of DVM-system capabilities to solve problems which use irregular grids”, *Trudy mezhdunarodnoy konferentsii «Superkomp'yuternyye dni v Rossii”* (26–27 sentyabrya 2016 g., g. Moskva), Izd-vo MGU, M., 2016, pp. 596–603 (in Russian). ↑32,33,55

[117] V. A. Bakhtin, V. A. Kryukov, N. V. Podderyugina, M. N. Pritula. “Implementation of parallel I/O in DVM-system”, *Trudy mezhdunarodnoy konferentsii «Superkomp'yuternyye dni v Rossii”* (26–27 sentyabrya 2016 g., g. Moskva), Izd-vo MGU, M., 2016, pp. 615–625 (in Russian). ↑32,33,55

[118] V. A. Bakhtin, A. S. Kolganov, V. A. Kryukov, N. V. Podderyugina, S. V. Polyakov, M. N. Pritula. “Extension of DVMH-model to work with irregular grids”, *Trudy mezhdunarodnoy nauchnoy konferentsii «Parallel'nyye vychislitel'nyye tekhnologii”*, PaVT'2016, Izdatel'skiy tsentr YuUrGU, Chelyabinsk, 2016, pp. 757. ↑32,33,55

[119] V. A. Bakhtin, O. F. Zhukova, N. A. Katayev, A. S. Kolganov, V. A. Kryukov, N. V. Podderyugina, M. N. Pritula, O. A. Savitskaya, A. A. Smirnov. “Automation of parallelization of software systems”, *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet”* (19–24 sentyabrya 2016 g., g. Novorossiysk), IPM im. M.V. Keldysha RAN, M., 2016, pp. 76–85. ↑32,33,55

[120] V. Krukov. “Automation of Programming at the M. V. Keldysh Institute of Applied Mathematics, Russian Academy of Sciences (KIAM RAS)”, *Proceedings 3rd International Conference on Computer Technology in Russia and in the Former Soviet Union*, SoRuCom 2014 (13–17 Oct. 2014, Kazan, Russia), 2015, ISBN 978-1-4799-1799-0, pp. 127–130. ↑32,33,55

[121] V. F. Aleksakhin, V. A. Bakhtin, O. F. Zhukova, A. S. Kolganov, V. A. Kryukov, I. P. Ostrovskaya, N. V. Podderyugina, M. N. Pritula, O. A. Savitskaya. “Parallelization in Fortran-DVMH language for Intel Xeon Phi coprocessor for NAS NPB3.3.1 tests”, *Trudy mezhdunarodnoy nauchnoy konferentsii «Parallel'nyye vychislitel'nyye tekhnologii”*, PaVT'2015, Izdatel'skiy tsentr YuUrGU, Chelyabinsk, 2015, pp. 19–30. ↑32,33,55

[122] V. A. Bakhtin, O. F. Zhukova, N. A. Katayev, A. S. Kolganov, N. V. Podderyugina, M. N. Pritula, O. A. Savitskaya, A. A. Smirnov, V. A. Kryukov. “Using the Internet to Teach Parallel Programming”, *Trudy XVII Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet”* (21–26 sentyabrya 2015 g., g. Novorossiysk), IPM im. M. V. Keldysha, 2015, pp. 26–33. ↑32,33,55

[123] V. F. Aleksakhin, V. A. Bakhtin, O. F. Zhukova, A. S. Kolganov, V. A. Kryukov, N. V. Podderyugina, M. N. Pritula, O. A. Savitskaya, A. V. Shubert. "Parallelization on GPUS of NPB 3.3 NAS tests on language FORTRAN DVMH", *Vestnik Ufimskogo gosudarstvennogo aviatsionnogo tekhnicheskogo universiteta*, **19**:1 (2015), pp. 240–250 (in Russian). ↑32,33,55

[124] V. F. Aleksakhin, V. A. Bakhtin, O. F. Zhukova, A. S. Kolganov, V. A. Kryukov, I. P. Ostrovskaya, N. V. Podderyugina, M. N. Pritula, O. A. Savitskaya. "Parallelization of NAS parallel benchmarks for Intel Xeon Phi coprocessor in Fortran-DVMH language", *Vestn. YuUrGU. Ser. Vych. matem. inform.*, **4**:4 (2015), pp. 48–63 (in Russian). ↑32,33,55

[125] V. A. Bakhtin, N. A. Konovalov, V. A. Kryukov. "Extension of the OpenMP Fortran language for programming GRID applications", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 23–28 sentyabrya 2002), Izd-vo MGU, M., pp. 273. ↑33

[126] V. Kasyanov, E. Kasyanova. "Methods and tools of parallel programming", International Conference Mathematical and Information Technologies (Vrnjacka Banja, Serbia–Budva, Montenegro, August 28–September 5, 2016), CEUR Workshop Proceedings, vol. **1839**, 2017, pp. 141–154. ↑33

[127] V. P. Gergel', R. G. Strongin. *Parallel Computing Basics for Multiprocessor Computing Systems*, 2 izd., NNGU, N. Novgorod, 2003, ISBN 5-85746-602-4, 184 pp. ↑33,36

[128] Zh. M. Abu-Khalil, S. A. Guda, B. Ya. Shteynberg. "Porting parallel programs while maintaining efficiency", *Otkrytyye sistemy. SUBD*, 2015, no. 4, pp. 18–19. ↑33,61

[129] Zh. M. Abu-Khalil, S. A. Guda, B. Ya. Shteynberg. "About high performance program portability", Trudy nauchnoy shkoly I. B. Simonenko, vol. **2**, Izd-vo Yuzhnogo federal'nogo universiteta, Rostov-na-Donu, 2015, ISBN 978-5-9275-1607-0, pp. 26–33. ↑33

[130] K. Goto, R. A. van de Geijn. "Anatomy of high-performance matrix multiplication", *ACM Trans. Math. Softw.*, **34**:3 (2008), 12, 25 pp. ↑34,50,52

[131] *Programming parallel processors*, ed. R.G. Babb, II, Addison-Wesley, 1989, ISBN 0201117215, 380 pp. ↑34

[132] V. A. Val'kovskiy. "Parallel execution of loops. Pyramid method", *Kibernetika*, 1983, no. 5, pp. 51–55. ↑35,51

[133] V. A. Val'kovskiy. "Parallel execution of loops. The parallelepiped method", *Kibernetika*, 1982, no. 2, pp. 51–62. ↑35,51

[134] D. Kulkurni, M. Stumm. *Loop and data transformations: a tutorial*, Technical Report CSRI-337, Computer Systems Research Institute, University of Toronto, 1993, 53 pp. ↑35

[135] K. E. Schauser. *Compiling dataflow into threads: efficient compiler-controlled multithreading for lenient parallel languages*, Technical Report No UCB/CSD-91-644, University of California, Berkley, 1991, 71 pp. ↑35

[136] N. Saboo, A. K. Singla, J. M. Under, L. V. Kale. "Emulating petaflops machines and blue gene", *Proceedings 15th International Parallel and Distributed Processing Symposium*, IPDPS 2001 (23–27 April 2001, San Francisco, CA, USA), 2001, ISBN 0-7695-0990-8, pp. 2084–2091. ↑35

[137] S. V. Torchigin. "Features of debugging applied and system tasks on a computing system with automatic resource allocation", *Tezisy dokladov Mezhdunarodnoy nauchnoy konferentsii «Intellektual'nyye i mnogoprotsessornyye sistemy"*, Izd-vo TRTU, Taganrog, 2003, pp. 55–57. ↑35

[138] A. Ye. Shiray. "System support for computing in combination with automatic resource allocation", *Tezisy dokladov Mezhdunarodnoy nauchnoy konferentsii «Intellektual'nyye i mnogoprotsessornyye sistemy"*, Izd-vo TRTU, Taganrog, 2003, pp. 54–55. ↑35

[139] V. S. Burtsev. "New approaches to assessing the quality of computing facilities", *Parallelizm vychislitel'nykh protsessov i razvitiye arkhitektury super·YeVM*, Neft' i gaz, M., 1997, pp. 28–40. ↑35

[140] V. Ye. Kotov, L. A. Cherkasova. "A network approach to describing the semantics of parallel systems and processes", Kibernetika i vychislitel'naya tekhnika, vol. **2**, Nauka, M., 1986, pp. 75–94. ↑35

[141] V. Ye. Kotov. *Petri nets*, Nauka. Glavnaya redaktsiya fiziko-matematicheskoy literatury, M., 1984, 159 pp. ↑35

[142] S. D. Bakhteyarov. *OKKAM programming language*, MNIIPU, M., 1989, 86 pp. ↑35

[143] A. S. Antonov. *Parallel programming using MPI technology*, Izd-vo MGU, M., 2004, ISBN 5-211-04907-1, 77 pp. URL ↑35

[144] A. A. Bukatov, V. N. Datsyuk, A. I. Zhegulo. *Programming multiprocessor computing systems*, Izd-vo OOO «TsVVR", Rostov-na-Donu, 2003, ISBN 5-94153-062-5, 208 pp. URL ↑35

[145] S. Nemnyugin, O. Stesik. *Parallel programming for multiprocessor computing systems*, BKhV-Peterburg, SPb., 2002, ISBN 5-94157-188-7, 400 pp. ↑35

[146] V. N. Kas'yanov. *Optimizing program transformations*, Nauka, M., 1988, ISBN 5-02-013778-2, 338 pp. ↑35,40

[147] K. Kasperskiy. *Program optimization technique. Efficient use of memory*, BKhV-Peterburg, SPb., 2003, ISBN 5-94157-232-8, 464 pp. ↑35,40,42

[148] L. Lamport. "The Coordinate Method for the parallel execution of DO loops", Sagamore Computer Conference on Parallel Processing, 1973, pp. 1–12. ↑36

[149] L. Lamport. "The parallel execution of DO loops", *Commun. ACM*, **17**:2 (1974), pp. 83–93. DOI ↑36,51

[150] V. A. Evstigneev, V. N. Kasyanov. "Optimizing transformations in paralleling compilers", *Programming and Computer Software*, **22**:6 (1996), pp. 279-290. ↑36,40

[151] A. P. Chernyayev. "Programming systems for high-performance computers", Itogi nauki i tekhniki. Vychislitel'nyye nauki, vol. **3**, VINITI AN SSSR, M., 1990, pp. 1–141. ↑36

[152] A. P. Chernyayev. "Software systems for vectorization and parallelization of FORTRAN programs for some vector conveyor computers (review)", *Programmirovaniye*, **17**:2 (1991), pp. 53–68. ↑36

[153] Yu. A. Frantsuzov. "Code parallelizing and software pipelining", *Programmirovaniye*, **18**:3 (1992), pp. 16–37 (in Russian). ↑36,40

[154] R. Allen, K. Kennedi. "Automatic translation of Fortran programs into vector form", *Vektorizatsiya programm: teoriya, metody, realizatsiya*, Mir, M., 1991, pp. 77–140. ↑36

[155] V. A. Serebryakov. "Cyclic software pipelining and translation of DO loops for tightly coupled multiprocessor systems", *Programmirovaniye*, **18**:3 (1992), pp. 54–60. ↑36

[156] U. Gupta, B. Reagen, L. Pentecost, M. Donato, Th. Tambe, A. M. Rush, Gu-Yeon Wei, D. Brooks. "MASR: A modular accelerator for Sparse RNNs", 28th International Conference on Parallel Architectures and Compilation Techniques, PACT 2019 (September 23-26, 2019, Seattle, WA, USA), pp. 1–14. arXiv:1908.08976~[eess.SP] ↑36

[157] Xiao Dong, Lei Liu, Peng Zhao, Guangli Li, Jiansong Li, Xueying Wang, Xiaobing Feng. "Acorns: a framework for accelerating deep neural networks with input sparsity", 28th International Conference on Parallel Architectures and Compilation Techniques, PACT 2019 (September 23-26, 2019, Seattle, WA, USA), pp. 178–191. ↑36

[158] D.-M. Loroch, F.-J. Pfreundt, N. Wehn, J. Keuper. "Sparsity in Deep Neural Networks — An Empirical Investigation with TensorQuant", Joint European Conference on Machine Learning and Knowledge Discovery in Databases (27 Aug. 2018), Communications in Computer and Information Science, vol. **967**, Springer, Cham, ISBN 978-3-030-14879-9, pp. 5–20. ↑36

[159] A. Guha, N. Vedula, A. Shriraman. "Deepframe: A Profile-Driven Compiler for Spatial Hardware Accelerators", 28th International Conference on Parallel Architectures and Compilation Techniques, PACT 2019 (September 23-26, 2019, Seattle, WA, USA), pp. 68–81. ↑37

[160] Yu. P. Kondratenko, V. V. Mokhor, S. A. Sidorenko. *Verilog-HDL for digital electronic circuit modeling and synthesis*, Uchebnoye posobiye, Izdatel'stvo NGTU, Nikolayev, 2002, 206 pp. ↑37

[161] I. V. Khakhanova. *Modeli i metody sistemnogo proyektirovaniya vychislitel'nykh struktur na kristallakh dlya tsifrovoy obrabotki signalov*, Dissertatsiya na soiskaniye uchenoy stepeni doktora tekhnicheskikh nauk, KhNUR·Ye, Khar'kov, 2008, 334 pp. ↑37

[162] A. M. Sergiyenko. *VHDL for designing computing devices*, ChP «Korneychuk", OOO «TID «DS", K., 2003, ISBN 966-7599-32-9, 208 pp. ↑37

[163] R. B. Shteynberg. "The parallelizing of recurrent loops with using of non-regular superpositions computations", *Izvestiya VUZov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2009, no. 2, pp. 16–18 (in Russian). ↑37

[164] R. B. Steinberg. "Mapping loop nests to multipipelined architecture", *Programming and Computer Software*, **36**:3 (2010), pp. 177–185. ↑37

[165] A. V. Kalyayev. "Programming virtual parallel problem-oriented supercomputers in the structure of universal supercomputers with massive parallelism", *Trudy mezhdunarodnoy nauchno-tekhnicheskoy konferentsii «Intellektual'nyye mnogoprotsessornyye sistemy"* (Taganrog, 1–5 sentyabrya 1999), pp. 27–39. ↑37,51

[166] A. V. Kalyayev, I. I. Levin. *Modularly scalable multiprocessor systems with structural and procedural organization of computations*, Yanus-K, M., 2003, 380 pp. ↑37,39

[167] I. I. Levin. "Modularly expandable multiprocessor computing system with structural and procedural organization of computations based on FPGA technologies", *Iskusstvennyy intellekt*, 2003, no. 4, pp. 446–453. ↑37

[168] A. V. Kalyayev, A. Yu. Artsatbanov, I. I. Itenberg. "Principles of building a family of high-performance orthogonal multiprocessor computing systems with

programmable architecture", Mnogoprotsessornyye vychislitel'nyye struktury, vol. **13(22)**, Taganrog, 1991, pp. 4–9. ↑37

[169] A. V. Kalyayev, I. I. Itenberg. "Organization of multisupertransputer computing systems based on macrocomputers with programmable architecture", *Tezisy dokladov IV Vsesoyuznogo seminara «Raspredelennaya obrabotka informatsii"* (Novosibirsk, 1991), pp. 4. ↑37

[170] A. V. Kalyayev. *Multiprocessor systems with programmable architecture*, Radio i svyaz', M., 1984, 240 pp. ↑37

[171] V. V. Korneyev. "Software configurability of the hardware structure", *Otkrytyye sistemy. SUBD*, 2007, no. 10. URL ↑37

[172] V. V. Korneyev. *Architecture of computing systems with programmable structure*, Nauka, Novosibirsk, 1985, 166 pp. ↑37

[173] E. V. Yevreinov, Yu. G. Kosarev. *High performance homogeneous general purpose computing systems*, Nauka, Novosibirsk, 1966. ↑37

[174] K. Bondalapati. *Modeling and mapping for dynamically reconfigurable hybrid architecture*, Ph.D. Thesis, University of Southern California, 2001, 194 pp. URL ↑37,38

[175] K. Bondalapati, V. K. Prasanna. "Reconfigurable computing systems", *Proceedings of the IEEE*, **90**:7 (2002), pp. 1201–1217. doi ↑37,38

[176] K. Compton, S. Hauck. "Reconfigurable computing: a survey of systems and software", *ACM Computing Surveys*, **34**:2 (2002), pp. 171–210. ↑37

[177] M. S. Jadzhak. "On a numerical algorithm of solving the cascade digital filtration problem", *Journal of Automation and Information Sciences*, **36**:6 (2004), pp. 23–34. doi ↑37

[178] M. S. Yadzhak, M. I. Tyutyunnyk. "An optimal algorithm to solve digital filtering problem with the use of adaptive smoothing", *Cybernetics and Systems Analysis*, **49**:3 (2013), pp. 449–456. doi ↑37

[179] A. V. Anisimov, M. S. Yadzhak. "Construction of optimal algorithms for mass computations in digital filtering problems", *Cybernetics and Systems Analysis*, **44**:4 (2008), pp. 465–476. doi ↑37

[180] M. G. Adigeyev, D. V. Dubrov, S. A. Lazareva, B. Ya. Shteynberg. "Experimental parallelizing compiler on a supercomputer with a structural organization of computations", *Tezisy dokladov vserossiyskoy nauchnoy konferentsii «Fundamental'nyye i prikladnyye aspekty razrabotki bol'shikh raspredelennykh programmnykh kompleksov"* (Novorossiysk, 21–26 sentyabrya 1998), Izd-vo MGU, M., 1998, pp. 101–108. ↑38

[181] D. Dubrov, A. Roshal. "Generating pipeline integrated circuits using C2HDL converter", East-West Design & Test Symposium (EWDTS 2013) (27-30 Sept. 2013, Rostov on Don, Russia), pp. 1–4. doi ↑38

[182] B. Ya. Steinberg, D. V. Dubrov, Yu. V. Mikhailuts, A. S. Roshal, R. B. Steinberg. "Automatic high-level programs mapping onto programmable architectures", *Proceedings of the 13th International Conference on Parallel Computing Technologies* (August 31–September 4, 2015, Petrozavodsk, Russia), Lecture Notes in Computer Science, vol. **9251**, Springer, Cham, ISBN 978-3-319-21908-0, pp. 474–485. doi ↑38

[183] B. Ya. Steinberg, A. P. Bugliy, D. V. Dubrov, Yu. V. Mikhailuts, O. B. Steinberg, R. B. Steinberg. "A project of compiler for a processor with programmable accelerator", 5th International Young Scientist Conference on Computational Science, YSC 2016 (26–28 October 2016, Krakow, Poland), *Procedia Computer Science*, **101** (2016), pp. 435–438. doi ↑38

[184] A. P. Bugliy, D. V. Dubrov, Y. V. Mikhailuts, B. Ya. Steinberg, R. B. Steinberg. "Developing a high-level language compiler for a computer with programmable architecture", *CEE-SECR'16: Proceedings of the 12th Central and Eastern European Software Engineering* (October 2016, Moscow, Russia), ISBN 978-1-4503-4884-3, 1–6 pp. ↑38

[185] Y. Liu, B. Schmidt. "SWAPHI: Smith-Waterman protein database search on Xeon Phi coprocessors", *ASAP 2014*. V. 1, 2014, pp. 184–185. ↑38,61

[186] D. V. Dubrov, A. S. Roshal', B. Ya. Shteynberg, R. B. Shteynberg. "Automatic mapping programs onto a processor with an FPGA accelerator", *Vestn. YuUrGU. Ser. Vych. matem. inform.*, **3**:2 (2014), pp. 117–121 (in Russian). ↑38

[187] K. Bondalapati, V. K. Prasanna. "Loop pipelining and optimization for run time reconfiguration", *IPDPS 2000: Parallel and Distributed Processing*, International Parallel and Distributed Processing Symposium, Lecture Notes in Computer Science, vol. **1800**, Springer, Berlin–Heidelberg, 2000, ISBN 978-3-540-67442-9, pp. 906–915. ↑38

[188] B. Y. Steinberg, A. P. Bagliy, Zh. M. Petrova, O. B. Steinberg. "Pipeline circuits to compute several expressions", *CEE-SECR'18: Proceedings of the 14th Central and Eastern European Software Engineering* (October 2018, Moscow, Russian Federation), ISBN 978-1-4503-6176-7, pp. 1–7. ↑38

[189] A. P. Bagliy, D. V. Dubrov, B. Ya. Shteynberg, R. B. Shteynberg. "Resource Reuse in Pipeline Computing", *Trudy XIX Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 18–23 sentyabrya 2017), IPM im. M.V. Keldysha, M., 2017, pp. 43–47. ↑38

[190] B. Ya. Shteynberg, A. P. Bagliy, D. V. Dubrov, Yu. V. Mikhayluts, O. B. Shteynberg, R. B. Shteynberg. "Classification of loops with one statement for executing on the processor with programmable accelerator", *Program Systems: Theory and Applications*, **8**:3, pp. 189–218 (in Russian). ↑38,42

[191] V. I. V'yukova, V. A. Galatenko, S. V. Samborskiy. "Using the CLP approach to combine software pipelining of the inner loop with the unwrapping of the outer loops when compiling the nest of nested loops", *Trudy chetvertoy mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2008 (Moskva, 27–29 oktyabrya 2008), pp. 1208–1220. ↑38

[192] S. S. Andreyev, S. A. Dbar, A. O. Latsis, Ye. A. Plotkina. "Programming system Autocode HDL and experience of its application for circuit implementation of numerical methods in FPGA", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet: masshtabiruyemost', parallel'nost', effektivnost"'* (Novorossiysk, 21–26 sentyabrya 2009), Izd-vo MGU, M., 2009, pp. 237. ↑38

[193] S. S. Andreyev, A. A. Davydov, S. A. Dbar, A. O. Latsis, Ye. A. Plotkina. *Technologies for the development of applied software for reconfigurable computing structures (FPGA)*, 2010 (Accessed 1.07.2020), 27 pp. ↑39

[194] D. J. Kuck, R. H. Kuhn, B. Leasure, M. Wolfe. "Dependence graphs and compiler optimizations", *POPL '81: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (Williamsburg, Va., Jan. 26–28), 1981, ISBN 978-0-89791-029-3, pp. 207–218. ↑39

[195] V. V. Petrenko. "Internal REPRISE representation of a parallelizing system", *Trudy chetvertoy mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2008 (27–29 oktyabrya 2008, Moskva, Rossiya). ↑39

[196] Ye. V. Alymova, A. P. Bagliy, D. V. Dubrov, R. A. Ibragimov, Yu. V. Mikhayluts, V. V. Petrenko, B. Ya. Shteynberg, R. B. Shteynberg, V. A. Yakovlev. "On the intermediate program representation for automatic generation of pipeline compute units", *Izvestiya vuzov. Severo-Kavkazskiy region. Tekhnicheskiye nauki*, 2017, no. 3, pp. 22–29 (in Russian). ↑39

[197] V. Petrenko, Ye. Metelitsa, R. Morylev, B. Shteynberg. "OPC-based program transformation training system "Parallel Programmer Simulator"", *Trudy Vserossiyskoy nauchnoy konferentsii pamyati A. L. Fuksmana «Yazyki programmirovaniya i kompilyatory"* (Rostov-na-Donu, 3–5 aprelya 2017, Yuzhnyy federal'nyy universitet), Izdatel'stvo Yuzhnogo federal'nogo universiteta, Rostov-na-Donu, 2017, pp. 198–201. ↑39

[198] Ye. Alymova, B. Shteynberg, A. Bagliy, D. Dubrov, V. Petrenko, R. Ibragimov, Yu. Mikhayluts, R. Shteynberg. "Intermediate representation of OPC programs for generating a pipeline calculator circuit", *Trudy Vserossiyskoy nauchnoy konferentsii pamyati A. L. Fuksmana «Yazyki programmirovaniya i kompilyatory"* (Rostov-na-Donu, 3–5 aprelya 2017, Yuzhnyy federal'nyy universitet), Izdatel'stvo Yuzhnogo federal'nogo universiteta, Rostov-na-Donu, 2017, pp. 38–41. ↑39

[199] K. Gufan, R. Morylev, V. Petrenko. "Visualization in the Open Parallelizing System", *Trudy Vserossiyskoy nauchnoy konferentsii «Nauchnyy servis v seti Internet"* (Novorossiysk, 18–22 sentyabrya 2006), pp. 58–61. ↑39,45

[200] Ye. N. Akimova, R. A. Gareyev. "Application of analytical modeling of matrix-vector multiplication on multicore processors", *Vestn. YuUrGU. Ser. Vych. matem. inform.*, **9**:1 (2020), pp. 69–82 (in Russian). ↑40,50,52

[201] R. Gareev, T. Grosser, M. Kruse. "High-performance generalized tensor operations: a compiler-oriented approach", *ACM Transactions on Architecture and Code Optimization*, **15**:3 (2018), pp. 1–27, 34. ↑40,52

[202] S. Muchnick. *Advanced Compiler Design Implementation*, 1 ed., Morgan Kaufmann, 1997, ISBN 978-1558603202, 888 pp. ↑40

[203] A. Akho, Dzh. Ul'man. *Parsing, Translation and Compilation Theory*. V. 1: *Sintaksicheskiy analiz*, Mir, M., 1978, 616 pp.; 243 pp. ↑40

[204] A. Akho, M. Lam, R. Seti, Dzh. Ul'man. *Compilers: Principles, Techniques, and Tools*, Per. s angl., 2-ye izd., Izdatel'skiy dom «Vil'yams", M., 2008, ISBN 978-5-8459-1932-8, 1184 pp. ↑40

[205] M. Griebl. *Automatic parallelization of loop programs for distributed memory architectures*, Habilitation, Passau University, 2004, 207 pp. ↑40,45,55

[206] D. J. Kuck, R. H. Kuhn, B. Leasure, M. Wolfe. "Dependence graphs and compiler optimizations", *POPL'81: Proceedings of the 8th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (Jan. 1981, Williamsburg, Va., USA), 1981, ISBN 978-0-89791-029-3, pp. 207–218. ↑40,45

[207] V. A. Yevstigneyev, S. V. Sprogis. "Vectorization of programs", *Vektorizatsiya programm: teoriya, metody, realizatsiya*, Mir, M., 1991, ISBN 5-03-001943-X, pp. 246–267. ↑40,42,43

[208] V. N. Kas'yanov, V. A. Yevstigneyev. *Graphs in programming: processing, visualization and application*, BKhV-Peterburg, SPb., 2003, ISBN 5-94157-184-4, 1104 pp. ↑40

[209] B. Ya. Shteynberg, Ye. V. Alymova, A. P. Bagliy, S. A. Guda, Ye. N. Kravchenko, R. I. Morylev, Z. Ya. Nis, V. V. Petrenko, I. S. Skiba, V. N. Shapovalov, O. B. Shteynberg. "Features of the implementation of parallelizing program transformations in

DVOR", *Trudy mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2010 (Moskva, 26–28 oktyabrya 2010, IPU RAN), pp. 787–854. ↑40

[210] R. Cytron, J. Ferrante, B. Rosen, M. N. Wegman, F. K. Zadeck. "Efficiently computing static single assignment form and the control dependence graph", *ACM Transactions on Programming Languages and Systems*, **13**:4 (1991), pp. 451–490. ↑40

[211] R. Ballance, A. Maccabe, K. Ottenstein. "The program dependence web: a representation supporting control-, data-, and demanddriven interpretation of imperative languages", *ACM SIGPLAN Notices*, **25**:6 (1990), pp. 257–271. ↑40

[212] P. Havlak. "Construction of thinned gated single-assignment form", *LCPC 1993: Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, vol. **768**, Springer, Berlin–Heidelberg, 1993, ISBN 978-3-540-57659-4, pp. 477–499. ↑40

[213] R. I. Morylev, V. N. Shapovalov, B. Ya. Shteynberg. "Symbolic analysis in dialog-based parallelization of programs", *Informatsionnyye tekhnologii*, 2013, no. 2, pp. 33–36 (in Russian). ↑40,46

[214] L. R. Gervich, E. N. Kravchenko, B. Y. Steinberg, M. V. Yurushkin. "Automatic program parallelization with block data distribution", *Num. Anal. Appl.*, **8**:1 (2015), pp. 35–45. ↑41,50,52

[215] Zh. M. Abu-Khalil, R. I. Morylev, B. Ya. Shteynberg. "Incremental modification of data structures of the internal representation of programs in the program parallelization system", *Sovremennyye problemy nauki i obrazovaniya*, 2013, no. 1, pp. 112–117. ↑41,51,52,61

[216] O. B. Shteynberg. "Classification of a programming loops with one assignment statement", *Izvestiya VUZov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2017, no. 1, pp. 52–58 (in Russian). ↑41,42

[217] O. B. Shteynberg. "The parallelizing of recurrent loops with using of non-regular superpositions computations", *Izvestiya VUZov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2009, no. 2, pp. 16–18 (in Russian). ↑41,42,47

[218] S. Ye. Sukhoverkhov, O. B. Shteynberg. "Automatic parallelizing of recurrent loops with stability control", *Informatsionnyye tekhnologii*, 2010, no. 1, pp. 40–45 (in Russian). ↑41,42,47

[219] O. B. Shteynberg. "Minimizing the number of node splitting in a problem of loops distribution", *Izvestiya VUZov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2011, no. 5, pp. 31–35 (in Russian). ↑41,42

[220] O. B. Steinberg. "Circular shift of loop body — programme transformation, promoting parallelism", *Vestn. YuUrGU. Ser. Matem. modelirovaniye i programmirovaniye*, **10**:3 (2017), pp. 120–132. ↑41,42

[221] O. B. Shteynberg, I. A. Ivlev. "Application of loops transformation "Retiming" to reduce the number of registers used", *Izvestiya vysshikh uchebnykh zavedeniy. Severo-Kavkazskiy region. Tekhnicheskiye nauki*, 2017, no. 3, pp. 76–80 (in Russian). ↑41,42

[222] V. A. Savel'yev, B. Ya. Shteynberg. *Parallelization of programs*, Izd-vo Yuzhnogo federal'nogo universiteta, Rostov-na-Donu, 2008, ISBN 978-5-9275-0547-0, 192 pp. ↑41

[223] V. V. Voyevodin, Vl. V. Voyevodin. *Parallel computing*, BKhV-Peterburg, SPb., 2002, ISBN 5-94157-160-7, 599 pp. ↑42,51

[224] D. Liu, Z. Shao, M. Wang, M. Guo, J. Xue. "Optimal loop parallelization for maximizing iteration-level parallelism", *CASES'09: Proceedings of the 2009 international conference on Compilers, architecture, and synthesis for embedded systems* (October 2009, Grenoble, France), 2009, ISBN 978-1-60558-626-7, pp. 67–76. ↑42

[225] D. Liu, Z. Shao, M. Wang, M. Guo, J. Xue. "Optimally maximizing iteration-level loop parallelism", *IEEE Transactions on Parallel and Distributed Systems*, **23**:3 (2012), pp. 564–572. ↑42

[226] P. Feautrier. "Dataflow analysis for array and scalar references", *International Journal of Parallel Programming*, **20**:1 (1991), pp. 23–53. ↑42,51

[227] V. Shamparov, A. Markin. "Structure Splitting for a compiler for microprocessors Elbrus", *CEE-SECR'19: Proceedings of the 14th Central and Eastern European Software Engineering Conference* (14–15 noyabrya 2019, Sankt-Peterburg, Rossiya). ↑42,49

[228] P. P. Chang, W.-W. Hwu. "Inline function expansion for compiling C programs", *ACM SIGPLAN Notices*, **24**:7 (1989), pp. 246–257. ↑43

[229] P. Zhao, J. N. Amaral. "To inline or not to inline? Enhanced inlining decisions", *LCPC 2003: Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, vol. **2958**, Springer, Berlin–Heidelberg, 2003, ISBN 978-3-540-21199-0, pp. 405–419. ↑43

[230] P. Andersson. *Evaluation of inlining heuristics in industrial strength compilers for embedded systems*, Examensarbete, Uppsala Universitet, 2009, 38 pp. ↑43

[231] A. V. Klimov, S. A. Romanenko. "Supercompilation: main principles and basic concepts", *Keldysh Institute preprints*, 2018, 111 (in Russian), 36 pp. ↑43

[232] D. V. Makoshenko. *Analiticheskoye predskazaniye vremeni ispolneniya programm i osnovannyye na nem metody optimizatsii*, Dissertatsiya na soiskaniye stepeni kandidata fiziko-matematicheskikh nauk po spetsial'nosti 05.13.11: matematicheskoye i programmnoye obespecheniye vychislitel'nykh mashin, kompleksov i komp'yuternykh setey, ISI SO RAN im. A. P. Yershova, Novosibirsk, 2011, 122 pp. ↑44

[233] Vl. V. Voyevodin. "Statistical estimates of the possibility of identifying the parallel structure of sequential programs", *Programmirovaniye*, **16**:4 (1990), pp. 44–54. ↑45

[234] Vl. V. Voyevodin. "Theory and practice of parallelism investigation in sequential programs", *Programmirovaniye*, **18**:3 (1992), pp. 38–54 (in Russian). ↑45

[235] S. A. Khozhaynova. "Statistical data on program parallelism", *Tezisy dokladov konferentsii «Smeshannyye vychisleniya i preobrazovaniya programm"* (Novosibirsk, Institut sistem informatiki i vychislitel'nyy tsentr AN SSSR, 27–29 noyabrya 1990), VTs SO RAN, Novosibirsk, 1991, pp. 237–242. ↑45

[236] R. N. Arapbayev. *Analiz zavisimostey po dannym: testy na zavisimost' i strategii testirovaniya*, Dissertatsiya na soiskaniye uchenoy stepeni kandidata fiziko-matematicheskikh nauk, ISI SO RAN, Novosibirsk, 2008, 116 pp. ↑45

[237] W. Pugh. "The Omega test: a fast and practical integer programming algorithm for dependence analysis", *Supercomputing'91: Proceedings of the 1991 ACM/IEEE conference on Supercomputing* (November 1991, Albuquerque, New Mexico, USA), 1991, ISBN 978-0-89791-459-8, pp. 4–13. ↑45

[238] S. V. Poluyan. *Analiz obrashcheniy programmy k pamyati v optimiziruyushchey rasparallelivayushchey sisteme*, Dissertatsiya na soiskaniye stepeni kandidata tekhnicheskikh nauk po spetsial'nosti 05.13.11: matematicheskoye i programmnoye obespecheniye vychislitel'nykh mashin, kompleksov i komp'yuternykh setey, SPbGU ITMO, Sankt-Peterburg, 2011, 147 pp. ↑45,46

[239] S. V. Poluyan. "Analysis of program accesses to memory in an optimizing parallelizing system", *Trudy XVIII Vserossiyskoy nauchno-metodicheskoy konferentsii Telematika'2011.* V. 2 (20–23 iyunya 2011, Sankt-Peterburg, Rossiya), SPbGU ITMO, SPb., 2011, pp. 328–329. ↑46

[240] S. V. Poluyan. "Profiling and its application in the interactive high-level optimizing parallelizer", *Trudy Mezhdunarodnoy superkomp'yuternoy konferentsii «Nauchnyy servis v seti Internet: superkomp'yuternyye tsentry i zadachi"* (20–25 sentyabrya 2010, Novorossiysk, Rossiya), Izd-vo MGU, M., 2010, pp. 652–653. ↑46

[241] B. Steinberg, A. Baglij, V. Petrenko, V. Burhovetckij, O. Steinberg, E. Metelica. "An analyzer for program parallelization and optimization", *ICAIT'2018: Proceedings of the 3rd International Conference on Applications in Information Technology* (November 2018, Aizu-Wakamatsu, Japan), ISBN 978-1-4503-6516-1, pp. 90–95. doi ↑47,60

[242] R. Arora, J. Olaya, M. Gupta. "A tool for interactive parallelization", *XSEDE'14: Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment* (July 2014, Atlanta, GA, USA), ISBN 978-1-4503-2893-7, pp. 1–8. doi ↑47

[243] L. R. Gervich, S. A. Guda, D. V. Dubrov, R. A. Ibragimov, E. A. Metelitsa, Yu. M. Mikhailuts, A. E. Paterikin, V. V. Petrenko, I. R. Skapenko, B. Ya. Steinberg, O. B. Steinberg, V. A. Yakovlev, M. V. Yurushkin. "How OPS (optimizing parallelizing system) may be useful for clang", *CEE-SECR'17: Proceedings of the 13th Central & Eastern European Software Engineering* (October 2017, St. Petersburg, Russia), 2017, ISBN 978-1-4503-6396-9, pp. 1–8. doi ↑47

[244] Z. Ya. Nis. "Ensuring static semantic correctness of programs when carrying out parallelizing and optimizing transformations", *Trudy chetvertoy mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2008 (27–29 oktyabrya 2008, Moskva, Rossiya). ↑47

[245] A. B. Bugerya. "Interactive debugging of parallel programs: distributed scheme of interacting components", *Programming and Computer Software*, **34**:3 (2008), pp. 154–159. ↑48

[246] M. Wolfe. "More iteration space tiling", *Supercomputing'89: Proceedings of the 1989 ACM/IEEE conference on Supercomputing* (November 1989, Reno, Nevada, USA), 1989, ISBN 978-0-89791-341-6, pp. 665–664. doi ↑48

[247] J. Xue, Q. Huang, M. Guo. "Enabling loop fusion and tiling for cache performance by fixing fusion-preventing data dependences", International Conference on Parallel Processing (ICPP'05) (14–17 June 2005, Oslo, Norway), 2005, pp. 107–115. doi ↑49

[248] K. Barton, J. Doerfert, H. Finkel, M. Kruse. *Loop Fusion, Loop Distribution and Their Place in the Loop Optimization Pipeline* (Accessed 1.07.2020), 20 pp. URL ↑49

[249] B. Ya. Shteynberg, O. B. Shteynberg, A. A. Vasilenko. "The loop fusion for data localization", *Program Systems: Theory and Applications*, **11**:3 (46) (2020), pp. 19–34 (in Russian). doi URL ↑49

[250] L. K. Eysymont, V. S. Gorbunov. "Towards an exascale supercomputer: results, directions, trends", MSKF 2012 (Moskva, 1 noyabrya 2012). ↑49

[251] A. Mycroft. “Programming language design and analysis motivated by hardware evolution”, *SAS 2007: Static Analysis*, International Static Analysis Symposium, Lecture Notes in Computer Science, vol. **4634**, Springer, Berlin–Heidelberg, 2007, ISBN 978-3-540-74061-2, pp. 18–33. ↑49

[252] A. I. Galushkin. “Development strategy of modern super-neurocomputers on the way to exaflop computing”, Prilozheniye k zhurnalu «Informatsionnyye tekhnologii”, 2012, 132 pp.↑49

[253] N. D. Morunov, D. L. Golovashkin. “Design features of block algorithms of FDTD-method implemented on a GPU using MATLAB”, *Computer Optics*, **43**:4 (2019), pp. 671–676 (in Russian). ↑49

[254] D. Kim, L. Renganarayanan, D. Rostron, S. Rajopadhye, M. M. Strout. “Multi-level tiling: M for the price of one”, *SC'07: Proceedings of the 2007 ACM/IEEE conference on Supercomputing* (November 2007, Reno, Nevada, USA), 2007, ISBN 978-1-59593-764-3, pp. 1–12. ↑50

[255] M. V. Yurushkin. “Double block data layout in high performance matrix multiplication algorithm”, *Programmnaya inzheneriya*, **7**:3 (2016), pp. 132–139 (in Russian). ↑50

[256] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, P. Wu, I. Yamazaki, A. Yarkhan, M. Abalenkovs, N. Bagherpour, S. Hammarling, J. Šístek, D. Stevens, M. Zounon, and S. D. Relton. “PLASMA: Parallel linear algebra software for multicore using OpenMP”, *ACM Transactions on Mathematical Software (TOMS)*, **45**:2 (2019), pp. 1–35, 16. ↑

[257] M. V. Yurushkin. “Block data layout automation in C language compiler”, *Programmnaya inzheneriya*, 2014, no. 6, pp. 16–18 (in Russian). ↑50,52

[258] L. R. Gervich, B. Ya. Shteynberg, M. V. Yurushkin. *Development of parallel programs with optimization of the use of the memory structure*, Izd-vo Yuzhnogo federal'nogo universiteta, Rostov-na-Donu, 2014, ISBN 978-5-9275-1500-4, 120 pp. ↑50,54

[259] B. Ya. Shteynberg. *Optimizing data placement in parallel memory*, Izd-vo Yuzhnogo federal'nogo universiteta, Rostov-na-Donu, 2010, ISBN 978-5-9275-0687-3, 255 pp.↑50,54,56

[260] V. V. Pakulev, V. V. Voyevodin. *Determining the Arcs of the Algorithm Graph*, Prepr. AN SSSR, Otd. vychisl. matematiki, No 228, Otdel vychislitel'noy matematiki AN SSSR, M., 1989, 22 pp.↑51

[261] V. V. Voyevodin. *Mathematical Foundations of Parallel Computing*, Izd-vo MGU, M., 1991, 345 pp.↑51

[262] P. Feautrier. “Some efficient solutions to the affine scheduling problem. I. One-dimensional time”, *International Journal of Parallel Programming*, **21**:5 (1992), pp. 313–347. ↑51

[263] P. Feautrier. “Some efficient solutions to the affine scheduling problem. II. Multidimentional time”, *International Journal of Parallel Programming*, **21**:6 (1992), pp. 389–420. ↑51

[264] P. Feautrier. “Parametric integer programming”, *RAIRO-Operations Research*, **22**:3 (1988), pp. 243–268. ↑51

[265] N. A. Likhoded. “Distribution of operations and data arrays over processors”, *Programming and Computer Software*, **29**:3 (2003), pp. 173–179.↑51

[266] S. A. Guda. "Estimates for the critical path in a lattice graph", *Trudy chetvertoy mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2008 (Moskva, 27–29 oktyabrya 2008), pp. 1253–1267. ↑51,57

[267] B. Ya. Shteynberg. "About the connection between the lattice graph and the data dependence graph", *Izvestiya VUZov. Severokavkazskiy region. Yestestvennyye nauki*, 2011, no. 5, pp. 29–31. ↑51

[268] A. Fernandez, J. M. Llaberia, M. Valero-Garcia. "Loop transformation using nonunimodular matrices", *IEEE Transactions on Parallel and Distributed Systems*, **6**:8 (1995), pp. 832–840. ↑51

[269] A. Hartono, Muthu Manikandan Baskaran, J. Ramanujam, P. Sadayappan. "DynTile: Parametric tiled loop generation for parallel execution on multicore processors", *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)* (19–23 April 2010, Atlanta, GA, USA), 2010. ↑52

[270] U. Bondhugula, A. Hartono, J. Ramanujam, P. Sadayappan. "A practical automatic polyhedral parallelizer and locality optimizer", *ACM SIGPLAN Notices*, 2008, pp. 101–113. ↑52

[271] A. Acharya, U. Bondhugula, A. Cohen. "Polyhedral auto-transformation with no integer linear programming", *Proceedings of 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI'18, ACM, New York, NY, USA, 2018, pp. 529–542. ↑52

[272] V. Bandishti, I. Pananilath, U. Bondhugula. "Tiling stencil computations to maximize parallelism", *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis* (Salt Lake City, Utah, USA, November 10–16, 2012). ↑52

[273] U. Bondhugula, A. Acharya, A. Cohen. "The Pluto+ algorithm: A practical approach for parallelization and locality optimization of affine loop nests", *ACM Trans. Program. Lang. Syst.*, **38**:3 (2016), 12, 32 pp. ↑52

[274] M. Wolfe, U. Banerjee. "Data dependence and its application to parallel processing", *International Journal of Parallel Programming*, **16**:2 (1987), pp. 137–178. ↑52

[275] S. B. Arykov, V. E. Malyshkin. "Asynchronous parallel programming system "Aspect"", *Vychislitel'nyye metody i programmirovaniye*, **9**:1 (2008), pp. 48–52 (in Russian). ↑52

[276] L. R. Gervich, B. Ya. Shteynberg, M. V. Yurushkin. "Exaflop Systems Programming", *Open Systems. DBMS*, 2013, no. 8, pp. 26–29 (in Russian). ↑52,54

[277] A. A. Korzh. "NPB UA benchmark scaling to thousands of Blue Gene/P cores using PGAS-like OpenMP extension", *Vychislitel'nyye metody i programmirovaniye*, **11** (2010), pp. 31–41 (in Russian). ↑52

[278] V. Agarwal, F. Petrini, D. Pasetto, D. Bader. "Scalable graph exploration on multicore processors", *SC'10: Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis* (13–19 Nov. 2010, New Orleans, LA, USA), 11 pp. ↑52

[279] B. Ya. Shteynberg. "Block-recurence matrix placement for Floyd algorithm", *Izvestiya VUZov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2010, no. 5, pp. 31–33 (in Russian). ↑52,54

[280] A. P. Bagliy, A. V.Boukhanovsky, B. Ya. Steinberg, R. B. Steinberg. "Baltic sea water dynamics model acceleration", *Vestn. YuUrGU. Ser. Matem. modelirovaniye i programmirovaniye*, **10**:1 (2017), pp. 113–124. ↑52

[281] S. G. Ammaev, L. R. Gervich, B. Y. Steinberg. "Combining parallelization with overlaps and optimization of cache memory usage", *PaCT 2017: Parallel Computing Technologies*, Lecture Notes in Computer Science, vol. **10421**, pp. 257–264. ↑52,54,60

[282] A. Yu. Perepelkina, V. D. Levchenko. "DiamondTorre algorithm for high-performance wave modeling", *Preprinty IPM im. M. V. Keldysha*, 2015, 018, 20 pp. ↑52

[283] B. A. Korneev, V. D. Levchenko. "Effective solving of three-dimensional gas dynamics problems with the Runge-Kutta discontinuous Galerkin method", *Comput. Math. Math. Phys.*, **56**:3 (2016), pp. 460–469. ↑52

[284] A. V. Zakirov, V. D. Levchenko, A. Yu. Perepelkina, Yasunari Zempo. "High performance FDTD code implementation for GPGPU supercomputers", *Preprinty IPM im. M. V. Keldysha*, 2016, 044, 22 pp. ↑52

[285] A. Yu. Perepelkina, V. D. Levchenko. "The DiamondCandy algorithm for maximum performance vectorized cross-stencil computation", *Preprinty IPM im. M. V. Keldysha*, 2018, 225, 23 pp. ↑52

[286] I. J. Bertolacci, C. Olschanowsky, B. Harshbarger, B. L. Chamberlain, D. G. Wonnacott, M. M. Strout. "Parameterized Diamond Tiling for Stencil Computations with Chapel parallel iterators", *ICS'15: Proceedings of the 29th ACM on International Conference on Supercomputing* (June 8-11, 2015, Newport Beach, CA, USA), pp. 197–206. ↑52

[287] Ki-Chang Kim. "Fine-grained parallelization of incomplete loop nests", *Programmirovaniye*, 1997, no. 2, pp. 52–66. ↑52

[288] S. L. Graham, M. Snir, C. A. Patterson. *Getting Up To Speed: The Future Of Supercomputing*, National Academies Press, 2005, ISBN 978-0309095020, 306 pp. ↑53

[289] Sh. Piper, Dzh. Pol, M. Skolt. "A new era in evaluating the performance of computer systems", *Otkrytyye sistemy. SUBD*, 2007, no. 9, pp. 52–58. ↑53

[290] V. V. Korneyev. "Problems of programming supercomputers based on multicore multithreaded crystals", *Trudy Vserossiyskoy superkomp'yuternoy konferentsii «Nauchnyy servis v seti Internet: masshtabiruyemost', parallel'nost', effektivnost"'* (Novorossiysk, 21–26 sentyabrya 2009), Izd-vo MGU, M., 2009, pp. 12–15. ↑54

[291] B. Ya. Shteynberg. "Block-affine data placements in a parallel memory", *Informatsionnyye tekhnologii*, 2010, no. 6, pp. 36–41 (in Russian). ↑54,56

[292] V. A. Savel'yev. "On optimization of parallel computations similar to direct method in the problem of heat conductivity for computer with distributed memory", *Izvestiya vuzov. Severo-Kavkazskiy region. Yestestvennyye nauki*, 2012, no. 4, pp. 12–14. ↑54

[293] U. Bondhugula. *Automatic distributed-memory parallelization and codegeneration using the polyhedral framework*, Technical report, ISc-CSA-TR-2011-3, 2011, 10 pp. ↑55

[294] T. Yuki, S. Rajopadhye. *Canonic multi-projection: memory allocation for distributed memory parallelization*, Technical Report CS11-106, Colorado State University, 12 pp. ↑55

[295] A. Syschikov, D. Rutkov. "Data allocation for parallel processing in distributed computing systems", 6-th Seminar Of Finnish-Russian University Cooperation in Telecommunication, 2009, 12 pp. ↑55

[296] S. M. Zotov, I. S. Golosov, Yu. B. Lifshits, V. D. Yemel'yanov, S. N. Orlov, A. B. Mnev. "Data distribution in automatically parallelizable programs for computers with distributed memory", *Informatsionnyye tekhnologii i informatsionnyye sistemy*, 1999, no. 1. ↑56

[297] P. Yunheung. *Compiling for Distributed Memory Multiprocessors Based on Access Region Analysis*, Thesis Ph.D in Computer Science, University of Illinois at Urbana-Champaign, 1997. ↑56

[298] Ye. A. Metlitskiy, V. V. Kaverznev. *Parallel memory systems: theory, design, application*, LGU, L., 1989, ISBN 9785288002472, 238 pp. ↑56

[299] A. V. Frolov. "Optimization of arrays placement in FORTRAN programs on multiprocessor computer systems", *Programmirovaniye*, 1998, no. 3, pp. 70–80. ↑56

[300] U. Banerjee, S. C. Chen, D. J. Kuck, R. A. Towle. "Time and parallel processor bounds for Fortran-like loops", *IEEE Transactions on Computers*, **C-28**:9 (1979), pp. 660–670. ↑57

[301] A. A. Samarskiy. *Introduction to numerical methods*, Nauka, M., 1987, ISBN 5-8114-0602-9, 280 pp. ↑57

[302] N. M. Yershov, A. M. Popov. "Optimization of parallel calculations with allowance for the architecture and operating speed of links in a computer system", *Vestnik Moskovskogo universiteta: Vychislitel'naya matematika i kibernetika*, 1993, no. 1, pp. 24–30. ↑58

[303] I. I. Levin, B. Ya. Shteynberg. "Comparative analysis of the efficiency of parallel programs for various architectures of parallel computers", *Iskusstvennyy intellekt*, 2001, no. 3, pp. 234–242. ↑58

[304] B. Ya. Shteinberg. "Parallelization of recurrent loops with conditional statements", *Autom. Remote Control*, **56**:9 (1995), pp. 1344–1350. ↑58

[305] V. Strassen. "Gaussian elimination is not optimal", *Numer. Math.*, **13**:4 (1969), pp. 354–356. ↑58

[306] A. Akho, Dzh. Khopkroft, Dzh. Ul'man. *Construction and analysis of computational algorithms*, Mir, M., 1979, ISBN 978-5-458-26604-8, 536 pp. ↑58

[307] L. A. Razborov. "P! = NP or Problem: A View from the 90s", *Matematicheskiye sobytiya 20-ogo Veka*, FAZIS, M., 2003, pp. 399–416. ↑58

[308] B. Ya. Shteynberg. "Dependence of the optimal distribution of the processor crystal area between memory and computational cores on the algorithm", *Trudy mezhdunarodnoy konferentsii «Parallel'nyye vychisleniya i zadachi upravleniya"*, PACO'2012 (24–26 oktyabrya 2012, IPU RAN, Moskva), pp. 99–108. ↑58

[309] V. M. Chernov. "Parallel machine arithmetic for recurrent number systems in non-quadratic fields", *Computer Optics*, **44**:2 (2020), pp. 274–281 (in Russian). ↑59

[310] V. V. Burkhovetskiy, B. Ya. Steinberg. "Parallelizing an exact algorithm for the traveling salesman problem", 6-th International Young Scientist Conference on Computational Science YSC 2017 (01–03 November 2017, Kotka, Finland), *Procedia Computer Science*, **119** (2017), pp. 97–102. ↑60

[311] V. V. Burkhovetskiy, B. Ya. Steinberg. "An exact parallel algorithm for traveling salesman problem", *CEE-SECR'17: Proceedings of the 13th Central & Eastern European Software Engineering* (20–22 Oct. 2017, St. Peterburg, Russian Federation), 5 pp. ↑60

[312] V. Burkhovetskiy, B. Shteynberg. "Strategy for using large jobs in parallel tree traversal", *Trudy Vserossiyskoy nauchnoy konferentsii pamyati A. L. Fuksmana «Yazyki programmirovaniya i kompilyatory"* (3–5 aprelya 2017, Yuzhnyy federal'nyy universitet, Rostov-na-Donu), Izdatel'stvo Yuzhnogo federal'nogo universiteta, Rostov-na-Donu, 2017, pp. 66–70. ↑60

[313] B. J. Steinberg, J. M. Abu-Khalil, M. G. Adigeyev, A. A. Bout, A. V. Kermanov, E. A. Pshenichnyy, G. V. Ramanchauskayte, A. P. Kroshkina, A. V. Gutnikov, N. S. Ponomareva, A. E. Panich, T. Shkurat. "A package of fast tools for genomic sequence analysis", *Int. J. Math. Models Methods Appl. Sci.*, **10** (2016), pp. 42–50. URL ↑61

[314] M. Farrar. "Striped Smith-Waterman speeds database searches six times over other SIMD implementations", *Bioinformatics*, **23**:2 (2007), pp. 156–61. doi ↑61

[315] J. Daily. "Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments", *BMC Bioinformatics*, **17** (2016), 81, 11 pp. doi ↑61

[316] Y. Liu, B. Schmidt, D. L. Maskell. "CUDASW++ 2.0: enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions", *BMC Research Notes*, **3**:1 (2010), 93. doi ↑61

[317] H. Lan, W. Liu, Y. Liu, B. Schmidt. "SWhybrid: a hybrid-parallel framework for large-scale protein sequence database search", 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS) (29 May–2 June 2017, Orlando, FL, USA), pp. 42–51. doi ↑61

Sample citation of this publication:

Boris Ya. Steinberg, Oleg B. Steinberg. "Program transformations as the base for optimizing parallelizing compilers". *Program Systems: Theory and Applications*, 2021, **12**:1(48), pp. 21–113. (*In Russian*). doi 10.25209/2079-3316-2021-12-1-21-113

URL http://psta.psiras.ru/read/psta2021_1_21-113.pdf