На правах рукописи

Штейнберг Олег Борисович

РАСПАРАЛЛЕЛИВАНИЕ ЦИКЛОВ ДОПУСКАЮЩИХ РЕКУРРЕНТНЫЕ ЗАВИСИМОСТИ

Специальность 05.13.11

Математическое и программное обеспечение

вычислительных машин, комплексов и компьютерных сетей

АВТОРЕФЕРАТ

диссертации на соискание ученой степени

кандидата физико-математических наук

Ростов-на-Дону

2014

Работа выполнена на факультете математики, механики и компьютерных наук федерального государственного автономного образовательного учреждения высшего профессионального образования «Южный федеральный университет».

Научный руководитель: заслуженный работник высшей школы РФ, кандидат физико-математических наук, профессор Ерусалимский Яков Михайлович.

Официальные оппоненты: Касьянов Виктор Николаевич, доктор физико-математических наук, профессор, заведующий лабораторией Федерального государственного бюджетного учреждения науки Института систем информатики им. А.П. Ершова Сибирского отделения Российской академии наук.

Макошенко Денис Валентинович, кандидат физико-математических наук, руководитель группы по разработке программного обеспечения ЗАО "Интел А/О".

Ведущая организация: Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный университет».

Защита диссертации состоится 15 мая 2014 года в 16.00 часов на заседании диссертационного совета Д 002.087.01 при федеральном государственном бюджетном учреждении науки Институте системного программирования Российской академии наук по адресу: 109004. Москва. Ул. Александра Солженицина, д. 25.

С диссертацией и авторефератом можно ознакомиться в библиотеке Федерального государственного бюджетного учреждения науки Института системного программирования Российской академии наук.

Автореферат разослан “14” апреля 2014 года.

Ученый секретарь
диссертационного совета
кандидат физ.-мат. наук.  /Зеленов С.В./

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы

Сферы применения высокопроизводительных вычислений постоянно пополняются все новыми областями. Такими областями являются: задачи биоинформатики, моделирования поведения различных видов техники, проектирования электростанций, прогнозирования погоды и изменения климата, задачи криптографии, анализа данных социальных сетей, цифровой обработки сигналов и многие другие. Также возникает все больше задач, для которых необходимы вычисления в реальном времени (мобильная связь, распознавание речи и визуальных образов и т.д.).

В наше время даже персональные компьютеры стали параллельными (многоядерными, многоканальными). По этой причине является целесообразным развитие технологий параллельного программирования.

Ручная оптимизация и ускорение программ делаются долго и дорого. Преодолеть эту проблему можно за счет автоматической оптимизации и распараллеливания.

По автоматической оптимизации и распараллеливанию программ было написано много теоретических работ и разработан ряд распараллеливающих оптимизирующих компиляторов. Среди теоретических работ по автоматическому распараллеливанию можно отметить статьи и монографии Л. Лэмпорта, Р. Аллэна, К. Кэннеди, М. Вольфа, С. Мучника, М. Лэм, А. Ахо, В. Вальковского, В. Воеводина, П. Фотрье. Такими компиляторами являются, например, *Clang/LLVM*, *Microsoft Visual C++*, *GCC*, *Intel C++ Compiler*.

Основным объектом распараллеливания являются программные циклы, поскольку в циклических участках программ сосредоточены большие объемы вычислений. При этом лишь небольшая их часть может быть распараллелена непосредственно. В связи с этим возникает мысль о поиске таких вспомогательных преобразований, после которых цикл можно частично или полностью распараллелить. Среди таких вспомогательных преобразований выделяются «разбиение цикла», «развертка цикла», «перестановка операторов», «введение временных массивов» и «растягивание скаляров».

«Развертка цикла» (*loop unrolling*) является эффективным распараллеливающим преобразованием на VLIW и суперскалярные архитектуры. Это преобразование применяется для использования векторизации на современных процессорах, поддерживающих технологии SSE, AVX и им подобных.

«Перестановка операторов» (*statement interchange*) используется для распараллеливания на VLIW архитектуру. Кроме того, это преобразование может быть использовано как вспомогательное для «разбиения цикла» и векторизации.

«Введение временных массивов» (*node splitting*) способствует «разбиению цикла» и векторизации.

«Растягивание скаляров» (*scalar expansion*) способствует «разбиению цикла» и векторизации. Подобные функции выполняют преобразования «экспансия массивов» (*array expansion*), «приватизация переменной» (*privatization*) и «повышение размерности массивов».

«Разбиение цикла» (*loop distribution, loop fission*) применяется при оптимизации использования кэш-памяти и для векторизации, в том числе частичной.

Преобразования «введение временных массивов» и «растягивание скаляров» ведут к дополнительному расходу памяти. Вопрос возможности минимизации этих расходов, до исследований представленных в диссертации, в литературе не рассматривался.

Среди рекуррентных циклов распараллеливаются, как правило, только циклы, вычисляющие сумму ряда. В связи с этим интересна задача автоматического распараллеливания рекуррентных циклов более сложного вида.

В случае, когда цикл не удается векторизовать целиком, к нему можно применить преобразование «разбиение цикла». Если при этом один из результирующих циклов можно будет векторизовать, то можно говорить о частичной векторизации. Вопрос частичной векторизации в литературе практически не освещается. Так, например, в статье Вольфа частичная векторизация описывается на примере, но алгоритм этого преобразования не приводится.

Исходя из вышесказанного видно, что для расширения множества программ, ускоряемых автоматическими оптимизирующими и распараллеливающими

преобразованиями, возникает необходимость разработки новых методов и алгоритмов, использующих комбинации вспомогательных преобразований циклов.

Объект исследования

Циклы последовательных программ и их преобразования к параллельному или частично параллельному коду.

Цель работы

Разработка новых алгоритмов преобразования программ, позволяющих существенно расширить класс последовательных программ, допускающих автоматическое распараллеливание.

Из сформулированной цели вытекают следующие **задачи**:

1. Разработка нового алгоритма «разбиения цикла» приводящего к распараллеливаемому виду более широкий класс программ, за счет использования вспомогательных преобразований.
2. Создание новых алгоритмов распараллеливания рекуррентных циклов с предварительным вычислением суперпозиций.
3. Разработка алгоритма, включающего одновременную векторизацию и распараллеливание циклов с линейной рекуррентной зависимостью.
4. Проведение численных экспериментов для оценки производительности разработанного алгоритма «распараллеливания рекуррентных циклов с предварительным вычислением суперпозиций».

Методы исследований

В процессе решения рассматриваемых задач использовались математические методы теории распараллеливающих/оптимизирующих преобразований программ, теории графов и линейной алгебры. Разработанные алгоритмы программно реализованы с использованием технологии объектно-ориентированного программирования. Эффективность разработанных алгоритмов оценивалась численными экспериментами.

Личный вклад

Автором лично были разработаны алгоритмы: «разбиения цикла с использованием вспомогательных преобразований», «распараллеливания рекуррентных циклов с опережающим вычислением коэффициентов», «распараллеливания рекуррентных циклов с предварительным вычислением суперпозиций», «распараллеливания циклов с линейной рекуррентной зависимостью» и «совместного применения распараллеливания и векторизации к циклам с линейной рекуррентной зависимостью». Алгоритм «разбиения цикла» создан с учетом минимизации количества применений вспомогательных преобразований, ведущих к дополнительным расходам памяти. Разработана программная реализация алгоритма «распараллеливания рекуррентных циклов с предварительным вычислением суперпозиций», учитывающая особенности архитектуры персональных компьютеров, для случая линейной и дробно-линейной рекуррентных зависимостей.

В рамках оптимизирующей распараллеливающей системы (OPS) автором были реализованы преобразования программ: «введение временных массивов», «растягивание скаляров», «разбиение цикла», «слияние цикла», «преобразование цикла к виду, допускающему векторизацию». На основе изложенных в диссертации алгоритмов были написаны соответствующие преобразования, также вошедшие в OPS.

Проведен ряд численных экспериментов.

Научная новизна

- Разработан новый алгоритм «разбиения цикла» использующий вспомогательные преобразования «введение временных массивов», «растягивание скаляров» и «перестановку операторов» и оптимизирующий объем используемой дополнительной памяти.
- Разработан новый алгоритм, включающий одновременную векторизацию и распараллеливание циклов с линейной рекуррентной зависимостью.
- Алгоритм «распараллеливания рекуррентных циклов с опережающим вычислением коэффициентов», обобщен на 2^K вычислительных устройств.

Практическая значимость

Разработанные алгоритмы распараллеливания могут быть использованы, как при автоматическом распараллеливании (в распараллеливающих компиляторах), так и при ручном и полуавтоматическом распараллеливании.

Результаты диссертации могут быть использованы разработчиками параллельного программного обеспечения и разработчиками распараллеливающих компиляторов.

Использование результатов работы

Результаты диссертации внедрены в программном обеспечении высокопроизводительной электроники ОАО «Ангстрем». Также часть результатов диссертации используются в учебном процессе мехмата Южного федерального университета в магистерской программе «Высокопроизводительные вычисления и технологии параллельного программирования» в спецкурсе «Параллельные вычисления и преобразования программ».

Степень достоверности и апробация результатов работы

Достоверность результатов подтверждается математической строгостью изложения алгоритмов, а также ускорением программ, установленным численными экспериментами.

Результаты, изложенные в диссертации, опубликованы в 9 научных работах, из которых 5 в журналах из перечня ВАК. Также, докладывались на семинарах и конференциях:

1. Региональная научно-практическая конференция молодых ученых и специалистов «Высокие информационные технологии» ВИНТП-2006, г. Ростов-на-Дону, 2006 г.
2. Научно-технический совет ОАО «НКБ ВС», г. Таганрог, март, 2008 г.
3. «IEEE East-West Design & Test» Symposium (EWDTS’09).). Moscow, Russia, September 18-21, 2009.
4. «Научный сервис в сети Интернет». г. Новороссийск, 20-26 сентября 2010 г.
5. На международной конференции «Параллельные вычисления и задачи управления» PACO’2010. (2010 г, Москва).

6. На всероссийской научной конференции по проблемам информатики «СПИСОК-2013», 23-26 апреля, 2013, матмех Санкт-Петербургского университета, г. Санкт-Петербург.
7. На постоянно действующем научном семинаре мехмата ЮФУ «Автоматическое распараллеливание программ».

Основные положения, выносимые на защиту

1. Алгоритм «разбиения цикла», использующий вспомогательные преобразования «введение временных массивов», «растягивание скаляров» и «перестановку операторов».
2. Алгоритм «распараллеливания циклов с линейной рекуррентной зависимостью» с постоянными коэффициентами.
3. Алгоритм «совместного применения распараллеливания и векторизации» циклов с линейной рекуррентной зависимостью с постоянными коэффициентами.
4. Алгоритмы распараллеливания рекуррентных циклов с предварительными вычислениями суперпозиций.

Структура и объем диссертации

Диссертация состоит из введения, трех глав, заключения, списка сокращений и списка литературы. Текст диссертации изложен на 114 страницах и содержит 29 примеров, 11 рисунков и 5 таблиц. Список литературы содержит 71 наименование.

Основное содержание работы

Во введении обосновывается актуальность темы работы, описывается состояние предметной области. Излагаются основные проблемы, возникающие при автоматическом распараллеливании программ, определяется цель работы, описываются основные идеи предлагаемых методов и задачи, которые они решают.

Первая глава носит вспомогательный характер. В ней описываются понятия, необходимые для изложения результатов диссертации. В **параграфе 1.1** приводятся определения из теории графов. В частности, определяются матрица достижимости, правильная нумерация и фактор-граф. **Параграф 1.2** посвящен

анализу информационных зависимостей в программе. Приводятся определения типов зависимостей, а также графа информационных связей и графа зависимостей по данным.

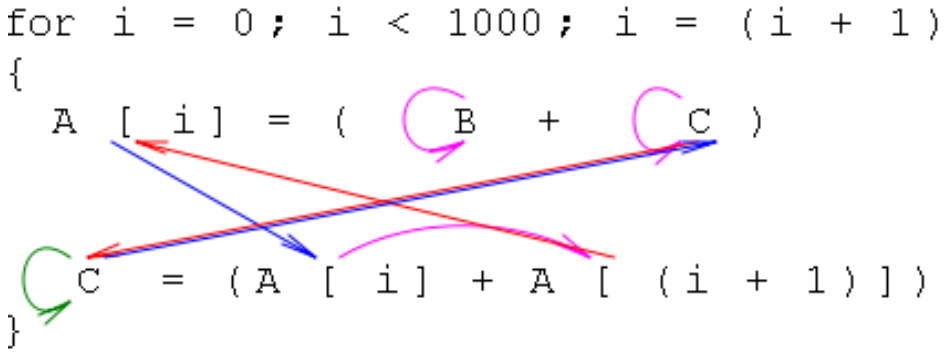

Рис.1 Граф информационных связей построенный с помощью OPS. чтение-чтение (in-in) – входная зависимость, запись-запись (out-out) – выходная зависимость, запись-чтение (out-in) – истинная зависимость, чтение-запись (in-out) – антизависимость.

В **параграфе 1.3** описывается Оптимизирующая распараллеливающая система, частью которой являются реализованные автором диссертации, алгоритмы и преобразования, описываемые в **главах 2** и **3**. **Параграф 1.4** содержит определения различных преобразований программ используемых в **главах 2** и **3**. Такими определениями, например, являются: «перестановка операторов», «расщепление цикла», «развертка» и «раскрутка» цикла. В **параграфе 1.5** приводятся два способа параллельного выполнения циклов, изложенные в известной статье Л.Лэмпорта «The parallel execution of DO loops». Один из этих способов подходит для параллельного выполнения цикла на компьютерах с архитектурой MIMD (в частности, для современных многоядерных процессоров), а другой на компьютерах с архитектурой SIMD (в частности, для векторных операций современных процессоров). Приводятся условия распараллеливания циклов на оба вида вычислительных архитектур. Лэмпортом описываются вспомогательные преобразования для распараллеливания, такие же, какие используются в данной работе для «разбиения цикла»: «перестановка операторов», «растягивание скаляров», «введение временных массивов».

Вторая глава посвящена преобразованию программ «разбиение цикла», способствующим ему вспомогательным преобразованиям и алгоритму «разбиения». В **параграфе 2.1** приводится определение того, в чем заключается

преобразование «разбиение цикла» и указаны введенные автором условия «разбиваемости». **Параграф 2.2** посвящен возможностям приведения цикла к «разбиваемому» виду. В этом параграфе описывается ряд вспомогательных преобразований, способствующих «разбиению». Приведены скриншоты программной реализации описываемых преобразований (программная реализация выполнена в рамках распараллеливающей системы OPS). В **параграфе 2.3** описывается алгоритм «разбиения цикла» использующий «перестановку операторов», «введение временных массивов», «растягивание скаляров» и «повышение размерности массива». Этот алгоритм включает минимизацию количества применений вспомогательных преобразований, вызывающих дополнительные расходы памяти. Такими преобразованиями являются «растягивание скаляров», «введение временных массивов» и «повышение размерности массива».

Рассмотрим пример применения алгоритма. К телу цикла приведенного на **рисунке 2.а** можно последовательно применить преобразования «растягивание скаляров», «перестановка операторов» и «введение временных массивов». В результате этих преобразований будет получен цикл, приведенный на **рисунке 2.б**. Полученный цикл можно разбить на четыре цикла содержащие по одному оператору присваивания. Каждый из результирующих циклов может быть векторизован.

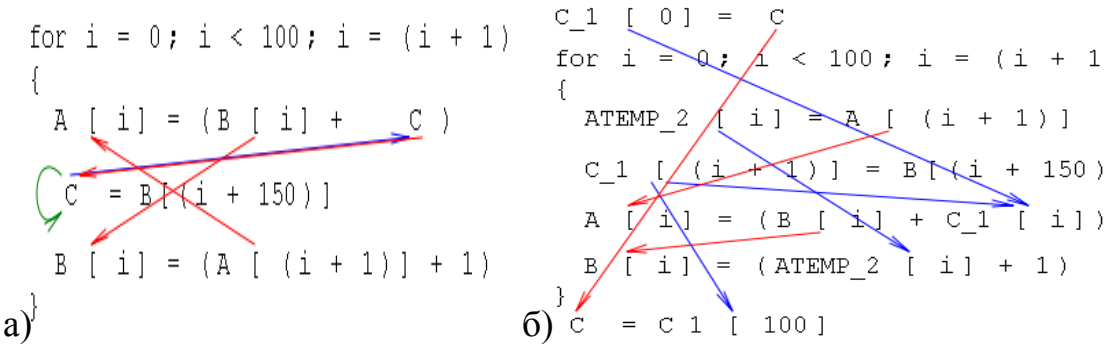

Рис. 2. На рисунке показаны эквивалентные циклы, с построенными (автоматически с помощью OPS) для их тел графами информационных связей.

В **параграфе 2.4** приведено преобразование «слияние циклов», обратное к «разбиению цикла».

В **третьей главе** приводятся разработанные в данной диссертации алгоритмы автоматической векторизации рекуррентных циклов. Эти алгоритмы реализованы в рамках Оптимизирующей распараллеливающей системы.

Автором разработаны тестовые программы для четырех известных компиляторов: Clang/LLVM версии 3.3 (релиз 17.06.2013), *Microsoft Visual C++* 2012 (релиз 15.08.2012), GCC 4.8.1 (релиз 31.05.2013), *Intel C++ Compiler* 13.1.3 (релиз 20.06.2013). Результаты тестирования выявили в этих компиляторах низкий уровень использования автоматической векторизации. Оказалось, что этими компиляторами не векторизуются циклы, предполагающие вспомогательные преобразования, «растягивание скаляров», «введение временных массивов». Не выполняется автоматическая векторизация и для часто встречающихся рекуррентных циклов.

Приведем циклы, которые не векторизуются ни одним из упомянутых компиляторов.

for(i = 0; i < SIZE; i++)
{
A[i] = B[i] + C[i];
C[i] = A[i+1] + B[i];
}
Может быть векторизован, после применения алгоритма «разбиения циклов» использующего вспомогательное преобразование «введение временных массивов»
	for(i = 0; i < SIZE; i++)
	{
	A[i] = B[i] + C[j];
	C[j] = A[i+1] + B[i];
	}
	Может быть векторизован, после применения алгоритма «разбиения циклов» использующего вспомогательные преобразования «повышение размерности массива» и «перестановка операторов»

for(i = 0; i < SIZE; i++)
{
A[i] = B[i]+C;
C = A[i+1]+B[i];
}
Может быть векторизован, после применения алгоритма «разбиения циклов» использующего вспомогательные преобразования «растягивание скаляров» и «перестановка операторов»
	for(int i = 0; i < SIZE; ++i)
	{
	A[i+2] = A[i+1] + A[i];
	}
	Векторизуем с помощью алгоритма описанного в **пункте 3.2.3**

Применение алгоритма разбиения цикла, описанного в **параграфе 2.3**, могло бы привести первые три цикла к виду, допускающему векторизацию. Последний цикл можно распараллелить и векторизовать одновременно алгоритмом **пункта 3.2.3**.

В **пункте 3.2.2** третьей главы рассмотрены методы автоматического распараллеливания рекуррентных циклов. Описан алгоритм преобразования цикла с линейной рекуррентной зависимостью с постоянными коэффициентами к виду, допускающему параллельное выполнение. Данный алгоритм является обобщением известных ранее методов и доведен до уровня программной реализации. Это преобразование сохраняет устойчивость вычислений. Устойчивость понимается в том смысле, что, если исходный рекуррентный цикл при малых изменениях входных данных давал малые изменения выходных данных, то таким же свойством обладает и преобразованный цикл (устойчивость доказана соавтором совместных исследований С.А. Суховерховым). Такое свойство устойчивости позволяет использовать это преобразование в распараллеливающих компиляторах.

В **пункте 3.2.3** описан алгоритм совместного применения распараллеливания и векторизации циклов с линейной рекуррентной зависимостью. Этот алгоритм является обобщением алгоритма из **пункта 3.2.2** и также обладает устойчивостью. Алгоритм состоит в преобразовании исходного

цикла к нескольким параллельным потокам, в каждом из которых применяется векторизация.

В **пункте 3.2.4** третьей главы рассмотрены методы распараллеливания рекуррентных циклов использующие предварительные вычислениями суперпозиций. Коэффициенты рекуррентной зависимости в данном пункте рассматриваются переменными в отличие от рекуррентных зависимостей, рассматриваемых в **пунктах 3.2.2** и **3.2.3**. Один из описываемых в этом пункте методов был известен ранее для распараллеливания вычислений на два вычислительных устройства. В данном параграфе этот метод обобщается на любое количество вычислительных устройств, которое является степенью двойки. Предложен алгоритм вычисления оптимальных параметров метода, который использует решение вспомогательной системы линейных уравнений с двухдиагональной матрицей.

На основе предложенных в **пункте 3.2.4** методов написаны параллельные программы, использующие технологию OpenMP и CUDA для двухядерных, четырехядерных процессоров и графического ускорителя соответственно. Эксперименты на двуядерном процессоре показывают ускорение на десятки процентов, а на четырехядерном процессоре преобразованная программа работает в два раза быстрее исходной.

С разработанными программами проведены численные эксперименты, результаты которых приводятся в **таблицах 1** и **2**.

Тесты проводились на двух компьютерах:

1) **Процессор**: Intel CoreTM i3-3227U 1.9GHz DDR3
2) **Процессор**: Intel Core i7-3820 3.60GHz DDR3

Таблица 1 Результаты тестирования быстродействия алгоритма «распараллеливания рекуррентных циклов с предварительным вычислением суперпозиций» в случае линейной рекуррентной зависимости.

	Последовательное выполнение	Выполнение 2-мя потоками	Выполнение 4-мя потоками	Используемые правые границы циклов
Intel i3	0.484124 сек.	0.310807 сек.	0.262585 сек.	N= 67108864
Intel i7	0.383944 сек.	0.255872 сек.	0.210152 сек.	

Значения столбца «последовательное выполнение» данной таблицы были получены при запуске цикла:

```
for(i = 1; i < N; i++)
{
  X[i] = A[i]*X[i-1]+B[i];
}
```

Таблица 2 Результаты тестирования быстродействия алгоритма «распараллеливания рекуррентных циклов с предварительным вычислением суперпозиций» в случае дробно-линейной рекуррентной зависимости.

	Последовательное выполнение	Выполнение 2-мя потоками	Выполнение 4-мя потоками	Используемые правые границы циклов
Intel i3	1.23998 сек.	0.792764 сек.	0.661197 сек.	N= 67108864
Intel i7	0.630645 сек.	0.404309 сек.	0.314366 сек.	

Значения столбца «последовательное выполнение» данной таблицы были получены при запуске цикла:

```
for(i = 1; i < N; i++)
{
  X[i] = (A[i]*X[i-1]+B[i]) / (C[i]*X[i-1]+D[i]);
}
```

На четырех ядрах процессора, эксперименты показывают ускорение почти в два раза. Это связано с особенностями архитектуры используемых компьютеров, а именно с числом каналов памяти. При использовании компьютеров с большим числом каналов памяти ускорение будет большим.

Численные эксперименты показали, что, используя модификацию алгоритма «распараллеливание рекуррентных циклов с опережающим вычислением суперпозиций», рекуррентные циклы можно вычислять на графических ускорителях быстрее, чем на персональных компьютерах.

Описанный алгоритм можно использовать для распараллеливания известного метода «прогонки», который используется для решения СЛАУ с трехдиагональной матрицей:

$$\left(\begin{array}{ccccccc|c} A_0 & B_0 & 0 & 0 & 0 & 0 & 0 & H_0 \\ C_1 & A_1 & B_1 & 0 & 0 & 0 & 0 & H_1 \\ 0 & C_2 & A_2 & B_2 & 0 & 0 & 0 & H_2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & C_{N-3} & A_{N-3} & B_{N-3} & 0 & H_{N-3} \\ 0 & 0 & 0 & 0 & C_{N-2} & A_{N-2} & B_{N-2} & H_{N-2} \\ 0 & 0 & 0 & 0 & 0 & C_{N-1} & A_{N-1} & H_{N-1} \end{array}\right)$$

Параграф 3.3 посвящен совместному применению преобразования «разбиение цикла» и распараллеливания циклов.

В этом параграфе рассмотрен метод частичной векторизации и распараллеливания циклов. Суть метода состоит в том, чтобы цикл, который не получается распараллелить или векторизовать непосредственно, разбивался на несколько циклов, некоторые из которых могли бы допустить параллельное или векторное выполнение.

В заключении подводятся итоги исследований, представленных в диссертации:

Разработан алгоритм «разбиения цикла», использующий вспомогательные преобразования «введение временных массивов», «растягивание скаляров» и «перестановку операторов».

Создан алгоритм «распараллеливания циклов с линейной рекуррентной зависимостью» с постоянными коэффициентами.

Приведен алгоритм «совместного применения распараллеливания и векторизации» циклов с линейной рекуррентной зависимостью с постоянными коэффициентами.

Получены алгоритмы «распараллеливания рекуррентных циклов с предварительными вычислениями суперпозиций».

Список публикаций

Статьи в журналах перечня ВАК:

1. Штейнберг О. Б. Распараллеливание рекуррентных циклов с нерегулярным вычислением суперпозиций. «Известия ВУЗов. Северокавказский регион. Естественные науки», №2, 2009 г., с. 18-21.

2. Штейнберг О. Б., Суховерхов С.Е. // Автоматическое распараллеливание рекуррентных циклов с проверкой устойчивости // «Информационные технологии», 2010, №1, с. 40-45.

3. Штейнберг Б.Я., Кравченко Е.Н., Морылев Р.И., Нис З.Я., Петренко В.В., Скиба И.С., Шаповалов В.Н., Штейнберг О.Б., Штейнберг Р.Б. Особенности реализации распараллеливающих преобразований программ в ДВОР. Известия ВУЗов. Приборостроение, т. 52, №10, 2011 г. с. 87-89.

4. Штейнберг О.Б. // Минимизация количества временных массивов в задаче разбиения циклов // «Известия ВУЗов. Северо-Кавказский регион. Естественные науки», 2011, №5 с. 31-35.

5. Духанов А. В., Болгова Е. В., Гервич Л. Р., Колпаков В. Г., Кравченко Е. Н., Курочкин И. И., Масленников Е. Д., Офёркин И. В., Рубцов А. О., Смирнов С. А., Штейнберг О. Б., Юрушкин М. В. Предметно-ориентированные технологии создания виртуальных рабочих пространств в среде облачных вычислений Clavire. Известия ВУЗов. Приборостроение, №5, 2013 г.

В других изданиях:

6. Штейнберг О.Б. // Распараллеливание рекуррентных циклов // «Высокие информационные технологии в науке и производстве» Тезисы докладов Ростов-на-Дону, Издательство Ростовского университета, 2006, с. 80.

7. Steinberg B., Abramov A., Alymova E., Baglij A., Guda S., Demin S., Dubrov D., Ivchenko A., Kravchenko E., Makoshenko D., Molotnikov Z., Morilev R., Nis Z., Petrenko V., Povazhnij A., Poluyan S., Skiba I., Suhoverkhov S., Shapovalov V., Steinberg O., Steinberg R. Dialogue-based Optimizing Parallelizing Tool and C2HDL Converter. Proceedings of IEEE East-West Design & Test Symposium (EWDTS'09). Moscow, Russia, September 18-21, 2009 (pp. 216 - 218)

8. Штейнберг Б.Я., Абрамов А.А., Алымова Е.В., Баглий А.П., Гуда С.А., Дубров Д.В., Кравченко Е.Н., Морылев Р.И., Нис З.Я., Петренко В.В., Полуян С.В., Скиба И.С., Шаповалов В.Н., Штейнберг О.Б., Штейнберг Р.Б., Юрушкин М. Диалоговый высокоуровневый автоматический распараллеливатель (ДВОР). Научный сервис в сети Интернет: Труды Всероссийской суперкомпьютерной конференции (20-26 сентября 2010 г., г. Новороссийск). М.: Изд-во МГУ, 2010., с. 71-75

9. Штейнберг Б.Я., Алымова Е.В., Баглий А.П., Гуда С.А., Кравченко Е.Н., Морылев Р.И., Нис З.Я., Петренко В.В., Скиба И.С., Шаповалов В.Н., Штейнберг О.Б. Особенности реализации распараллеливающих преобразований программ в ДВОР. РАСО'2010/ Труды международной конференции «Параллельные вычисления и задачи управления». М., 26-28 октября 2010 г., ИПУ РАН, с.787-854

Свидетельство о государственной регистрации программ для ЭВМ:

10. Штейнберг Б.Я., Штейнберг Р.Б., Морылев Р.И., Петренко В.В., Полуян С.В., Штейнберг О.Б., Баглий А.П., Нис З.Я., Скиба И.С., Юрушкин М.В., Шаповалов В.Н., Алымова Е.В., Кравченко Е.Н., Гуда С.А. «Диалоговый высокоуровневый оптимизирующий распараллеливатель программ» Свидетельство о государственной регистрации программы для ЭВМ №2011617205 от 15 сентября 2011г.