ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

СРЕДСТВА АРХИТЕКТУРНО-ОРИЕНТИРОВАННОЙ ОПТИМИЗАЦИИ ВЫПОЛНЕНИЯ ПАРАЛЛЕЛЬНЫХ ПРОГРАММ ДЛЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ С МНОГОУРОВНЕВЫМ ПАРАЛЛЕЛИЗМОМ

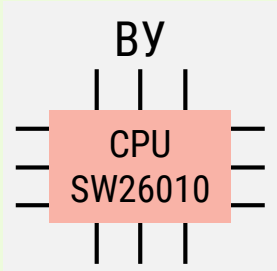
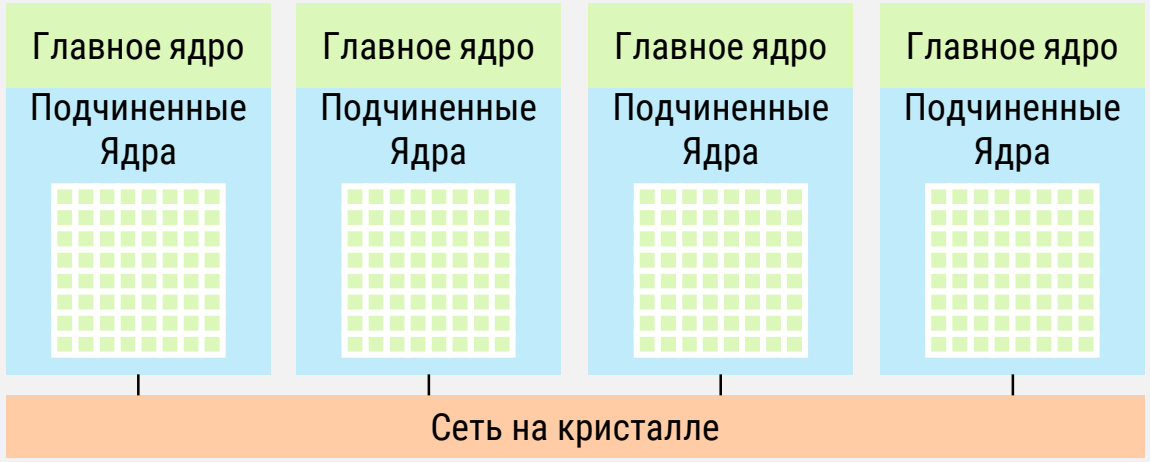
Кулагин Иван Иванович

Диссертация на соискание ученой степени кандидата технических наук
Специальность: 05.13.15 «Вычислительные машины, комплексы и компьютерные сети»

Научный руководитель –
доктор технических наук, доцент,
Курносов Михаил Георгиевич

г. Новосибирск, 15 февраля 2018 г.

АКТУАЛЬНОСТЬ ТЕМЫ ИССЛЕДОВАНИЯ

	Sunway TaihuLight (№ 1 TOP500)	**Titan – Cray XK7** (№ 4 TOP500)
Параллелизм уровня вычислительных узлов	40 960 × 	18 688 × (1 – AMD Opteron 6274) (1 – NVIDIA Tesla K20)
Параллелизм потоков на уровне ядер	1 ВУ = 1 CPU (SW26010) = 260 ядер = (4 × [64 + 1]) ядер 	1 ВУ = 16 ядер CPU AMD Opteron 6274 + 2 496 ядра NVIDIA Tesla K20 (SIMT – Single Instruction Multiple Threads)
Параллелизм команд на уровне АЛУ	Процессор на базе архитектуры **Alpha / SPARC ?** Ядро микроархитектуры **ShenWei** (8 АЛУ)	AMD Opteron 6274: Микроархитектура **Buildozer** (8 АЛУ)
Параллелизм уровня данных	Подчиненные ядра SW26010: поддержка векторных SIMD-команд	Ядра AMD Opteron 6274: поддержка векторного расширения **AVX**

ЦЕЛЬ РАБОТЫ И ЗАДАЧИ ИССЛЕДОВАНИЯ

Целью работы является разработка и исследование средств архитектурно-ориентированной оптимизации выполнения параллельных программ для ВС с многоуровневым параллелизмом.

Задачи исследования:

1. *Для ВС с многоуровневым параллелизмом разработать средства оптимизации циклического доступа к информационным массивам в параллельных программах на базе модели разделенного глобального адресного пространства.*
2. *Для многопроцессорных вычислительных узлов с общей памятью разработать и исследовать методы сокращения времени выполнения транзакционных секций многопоточных программ в модели программной транзакционной памяти.*
3. *Разработать средства анализа эффективности векторизации циклов на архитектурах процессоров с короткими векторными регистрами.*
4. *Разработать пакет программ оптимизации функционирования ВС и использования их многоуровневого параллелизма для решения параллельных задач.*

СТРУКТУРА ДИССЕРТАЦИИ

Введение

Глава 1. Вычислительные системы с многоуровневым параллелизмом

Глава 2. Модель разделенного глобального адресного пространства

Глава 3. Оптимизация выполнения программ на многопроцессорных ВС с общей памятью

Глава 4. Мультикластерная ВС

Заключение

Литература

Приложения

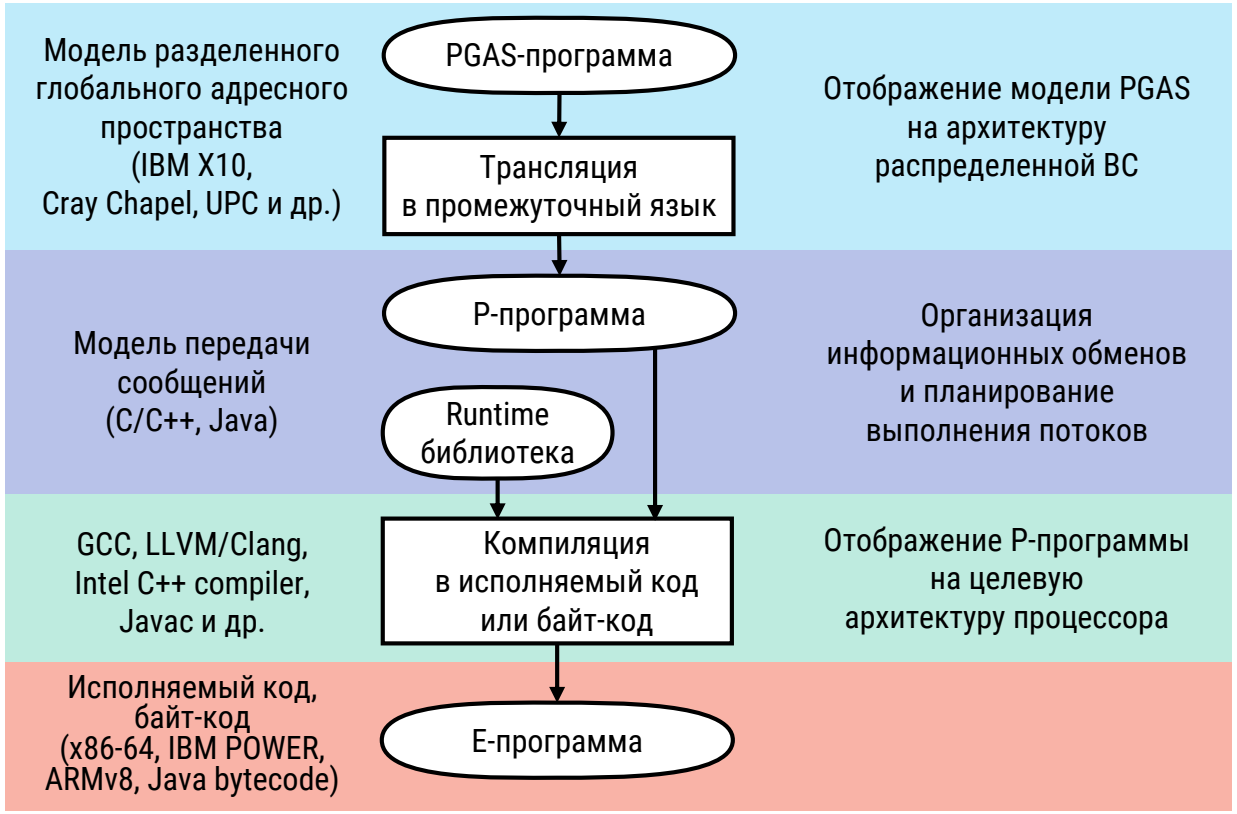
Работа основана на результатах ведущей научной школы в области анализа и организации функционирования распределенных ВС
(*руководитель – чл.-корр. РАН В.Г. Хорошевский*)

Модель разделенного глобального адресного пространства (Глава 2)

МОДЕЛЬ РАЗДЕЛЕННОГО ГЛОБАЛЬНОГО АДРЕСНОГО ПРОСТРАНСТВА

Отображение модели разделенного глобального адресного пространства (PGAS) на ресурсы ВС

Процесс компиляции PGAS-языков



[1] D. Callahan, B. Chamberlain, and H. Zima. **The Cascade High Productivity Language //** In International Workshop on High-Level Parallel Programming Models and Supportive Environments (HIPS 2004), pages 52–60, 2004.

[2] P. Charles, C. Donawa, K. Ebcioglu, C. Grothoff, A. Kielstra, C. Praun, V. Saraswat, and V. Sarkar. **X10: An Object-oriented approach to non-uniform Clustered Computing** // In 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA 2005), pages 519–538, 2005.

ЗАДАЧА ТРАНСФОРМАЦИИ КОНСТРУКЦИЙ ЦИКЛИЧЕСКОЙ ПЕРЕДАЧИ ПОТОКА УПРАВЛЕНИЯ ПОДЧИНЕННЫМ ЭЛЕМЕНТАРНЫМ МАШИНАМ

Дано: PGAS-программа с конструкцией циклической передачи потока управления подчиненным элементарным машинам (ЭМ)

В конструкции присутствует цикл из r итераций, на каждой итерации i:

Передается поток управления подчиненным ЭМ из множества M

Количество $m = |M|$ передач потока управления ЭМ

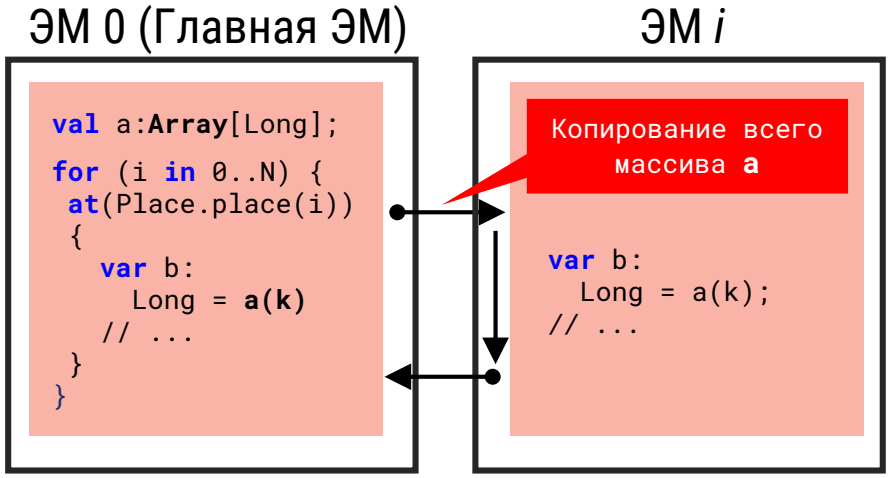
Код внутри конструкции удовлетворяет требованиям:

1. В теле конструкции выполняются операции над элементами l массивов $a_0,...,a_{l-1}$
2. Тело конструкции представлено функциями $f_0(a_0),...,f_{l-1}(a_{l-1})$, где функция $f_i(a_i)$ содержит операции, выполняемые над элементами массива a_i
3. Массивы $a_0,...,a_{l-1}$ хранятся в памяти главной ЭМ (ЭМ с номером 0)

Требуется: Разработать алгоритм преобразования циклических конструкций передачи потока управления ЭМ из модели PGAS в модель передачи сообщений.

Пример конструкции:

```
1   val a:Array[Long] =
2     new Array[Long](length, (i:Long) => i);
3   val M = Place.places();
4   ...
5       for (i in 0..(r – 1)) {
6           for (j in M) {
7               at (j) async {
8                   foo(A);
9                   ...
10              }
11          }
12      }
```

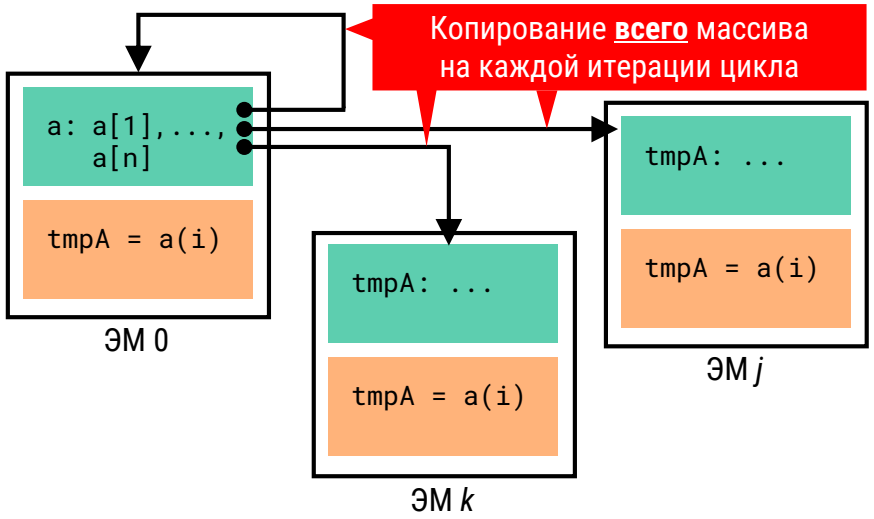


- Массив ***a*** хранится в памяти ЭМ 0
- Для выполнения тела конструкции массив ***a*** должен храниться в памяти ЭМ 1
- Доставку массива ***a*** из памяти ЭМ 0 в память ЭМ 1 выполняет runtime-система при помощи коммуникационных операций
- **Коммуникационные операции генерируются компилятором PGAS-языка**

Объем передаваемых данных определяется эффективностью алгоритма трансформации конструкции циклической передачи потока управления!

АЛГОРИТМЫ ТРАНСФОРМАЦИИ КОНСТРУКЦИЙ ЦИКЛИЧЕСКОЙ ПЕРЕДАЧИ ПОТОКА УПРАВЛЕНИЯ ПОДЧИНЕННЫМ ЭМ

Стандартный алгоритм *By-Iterative Copying* [1]



- На каждой итерации происходит обращение к коммуникационным операциям
- На каждой итерации ЭМ 0 копирует весь массива ***a*** в память ЭМ *j* и *k*

– Высокие накладные расходы на передачу массивов, т.к. происходит избыточное копирование всего массива

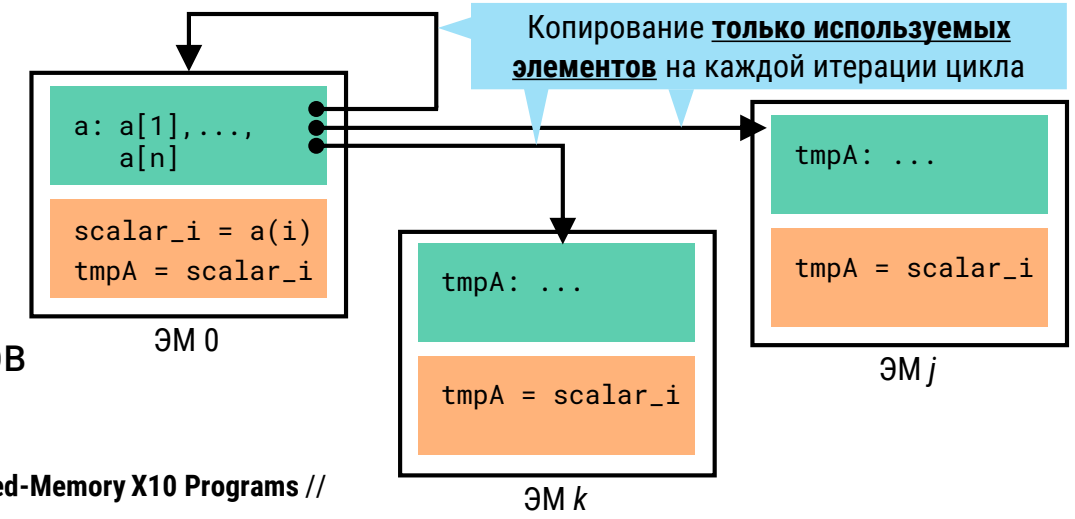
\+ Алгоритм применим для всех видов конструкций

Известный алгоритм *Scalar Replacement* [1]

- На каждой итерации происходит обращение к коммуникационным операциям
- На каждой итерации ЭМ 0 копирует только используемые элементы массива ***a*** в память ЭМ *j* и *k*

– Узкая область применения. Не применим, если на этапе компиляции неизвестно множество используемых элементов

\+ Копируются только используемые элементы массивов



[1] R. Barik, J. Zhao, D. Grove, I. Peshansky, Z. Budimlic, and V. Sarkar. **Communication Optimizations for Distributed-Memory X10 Programs** // In IEEE International Parallel and Distributed Processing Symposium, pages 1–13, 2011.

Алгоритм Array Preload опережающего копирования массивов

Разработан алгоритм *Array Preload*

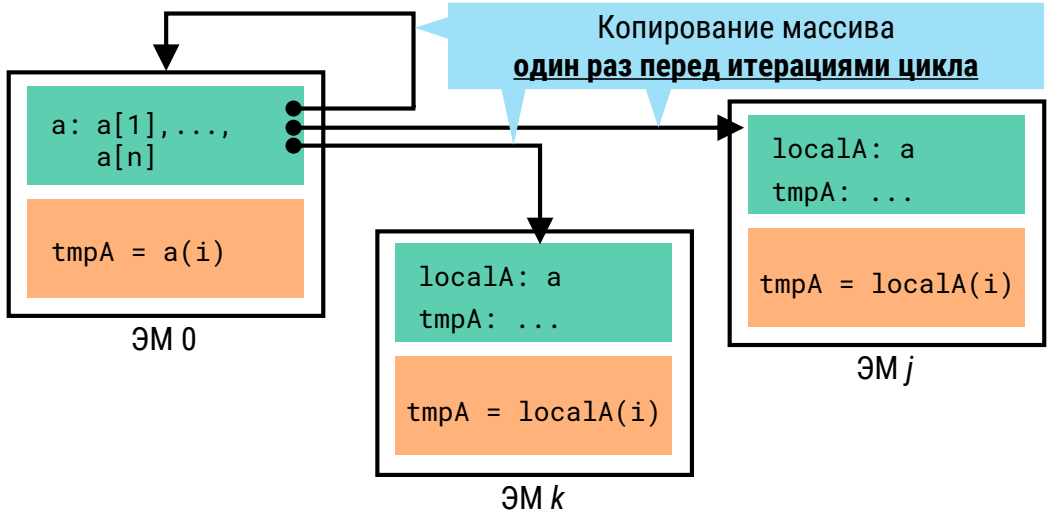
- Формирует пролог цикла для доставки требуемых элементов в память подчиненных ЭМ

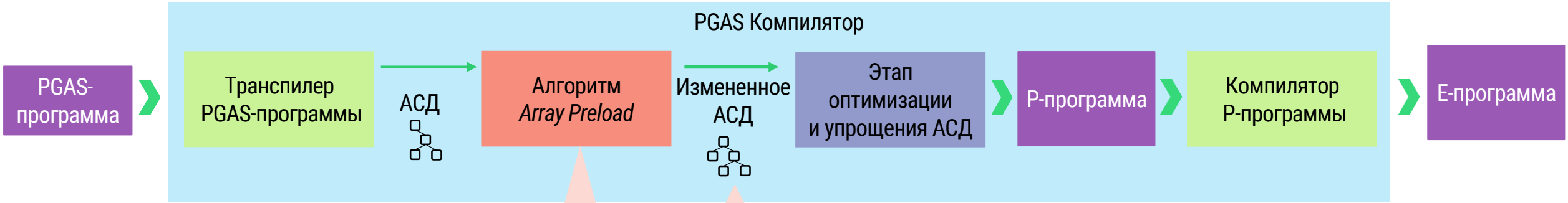
– Значительная трансформация цикла

\+ Отсутствует избыточное копирование массивов

\+ Применим в случаях, когда множество индексов используемых элементов массивов неизвестно

\+ Отсутствует функциональная зависимость объема передаваемых данных от числа итераций цикла





1. Построение пролога цикла
2. Создание локальных копий массивов $a_0, \ldots, a_{l-1}$ в памяти подчиненных ЭМ
3. Вставка обращений к элементам локальных массивов $a'_0, \ldots, a'_{l-1}$ в тело конструкции

1. Добавлен пролог цикла
2. Изменен доступ к массивам $a_0, \ldots, a_{l-1}$ в теле конструкции

ИССЛЕДОВАНИЕ АЛГОРИТМОВ

В моделях параллельных вычислений **построены** оценки времени выполнения кода, генерируемого алгоритмами (Модель Hockney)

By-Iterative Copying: $t = rm(\alpha + \beta b \sum_{i=0}^{l-1} s_i)$

Scalar Replacement: $t = rm(\alpha + \beta bq)$

Array Preload: $t = m(\alpha + \beta b \sum_{i=0}^{l-1} s_i)$

s_i – размер массива a_i m – количество подчиненных ЭМ

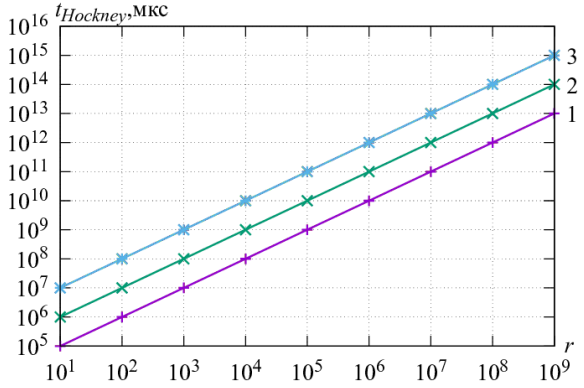
b – размер одного элемента массива r – число итераций в цикле

q – количество используемых элементов массивов в теле конструкции

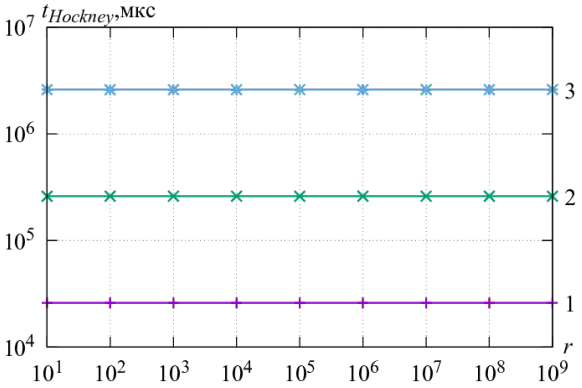
Зависимость времени выполнения от числа итераций

Сеть InfiniBand QDR ($\alpha = 1$ мкс; $\beta = 2 * 10^{-4}$ мкс)

Scalar Replacement

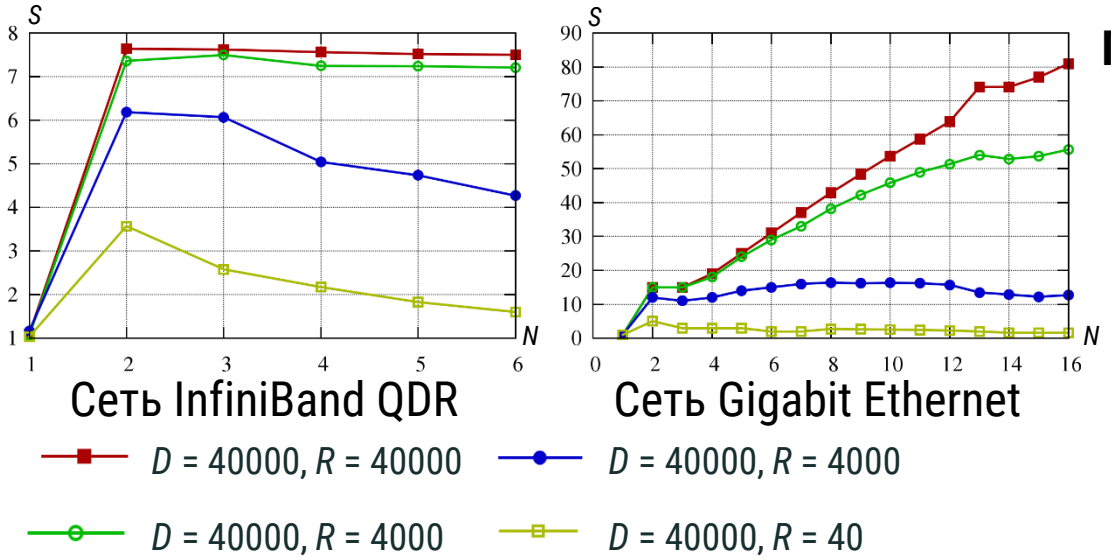


Array Preload



1 – $m = 10^4$; 2 – $m = 10^5$; 3 – $m = 10^6$

Зависимость ускорения от числа ЭМ



Проведено экспериментальное исследование алгоритма *Array Preload*

Тестовая программа:
Случайный доступ к элементам удаленного массива
Размер массива (*D*);
Количество итераций (*R*);
Количество ЭМ (*N*).

Условия экспериментов:

16 узлов: 2 x Quad-Core Intel Xeon E5420, Gigabit Ethernet, InfiniBand QDR
PGAS Компилятор: IBM X10 2.4, C++ backend – GCC 4.8.3.

Оптимизация выполнения программ на многопроцессорных ВС с общей памятью (Глава 3)

ПРОГРАММНАЯ ТРАНЗАКЦИОННАЯ ПАМЯТЬ

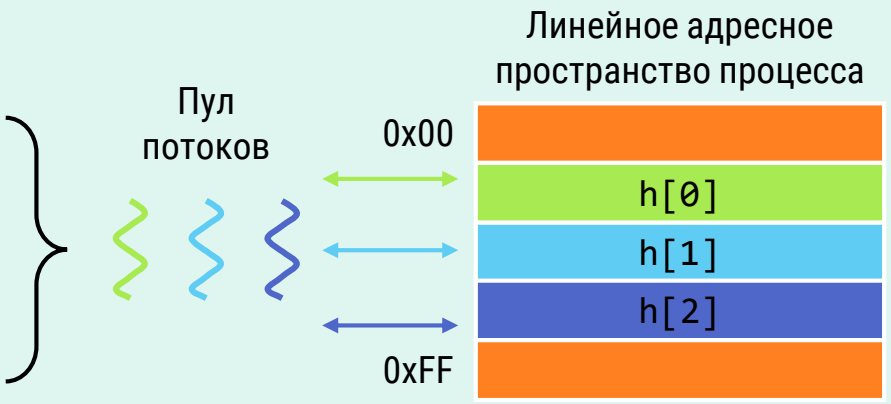
Уровень многопроцессорного вычислительного узла с общей памятью:

- Используется модель многопоточного программирования
- Актуальны задачи сокращения времени доступа потоков параллельных программ к разделяемым структурам данных

Синхронизация на основе механизма блокировок

Код, выполняющийся множеством потоков

```
function hash_table_add(h, key, value)
    lock_acquire ()
      i = hash(key)
      list_add_front(h[i], key, value)
    lock_release ()
end function
```



Механизм блокировок

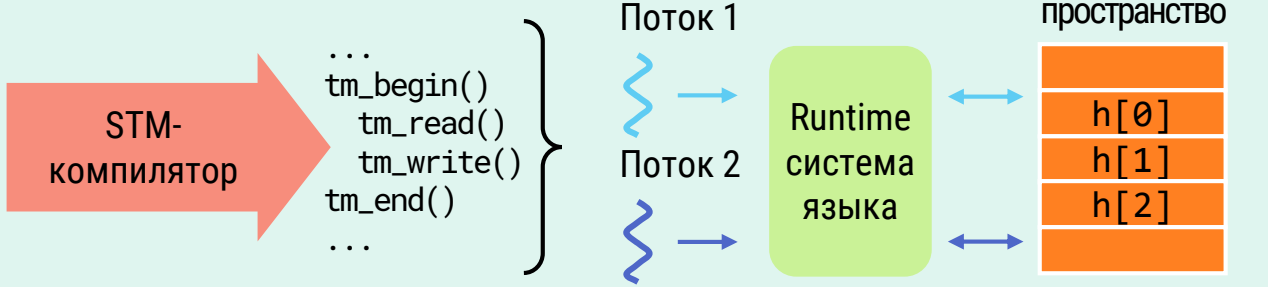
- Каждый поток добавляет узел в отдельный связный список h[i]
- Часто случается так, что блокировка потоков не требуется

Необходимо защищать области памяти, а не участки кода

Синхронизация при помощи транзакционной памяти (software transactional memory – STM)

Код, выполняющийся множеством потоков

```
function hash_table_add(h, key, value)
    transactional_section {
      i = hash(key)
      list_add_front(h[i], key, value)
    }
end function
```



Транзакционная память:

- Потоки не блокируются
- **Защищается разделяемая память!**

[1] M. Herlihy, J. E. B. Moss. **Transactional memory: architectural support for lock-free data structures //** Proceedings of the 20th annual international symposium on Computer architecture (ISCA '93). Volume 21, Issue 2, May 1993.
[2] N. Shavit, D. Touitou. **Software Transactional Memory** // In PODC'95: Proceedings of the 14th annual ACM symposium on Principles of distributed computing, New York, NY, USA, Aug. 1995. ACM, 204–213.

ЗАДАЧА О ПРЕДОТВРАЩЕНИИ ВОЗНИКНОВЕНИЯ ЛОЖНЫХ КОНФЛИКТОВ

При выполнении транзакционных секций runtime-система отслеживает возникновение **состояний гонок за данными**

- Внутри транзакционной секции выполняется запись в глобальную переменную ***shared_val***

```
...
transactional_section {
    shared_val = 5
}
...
```

STM-компилятор

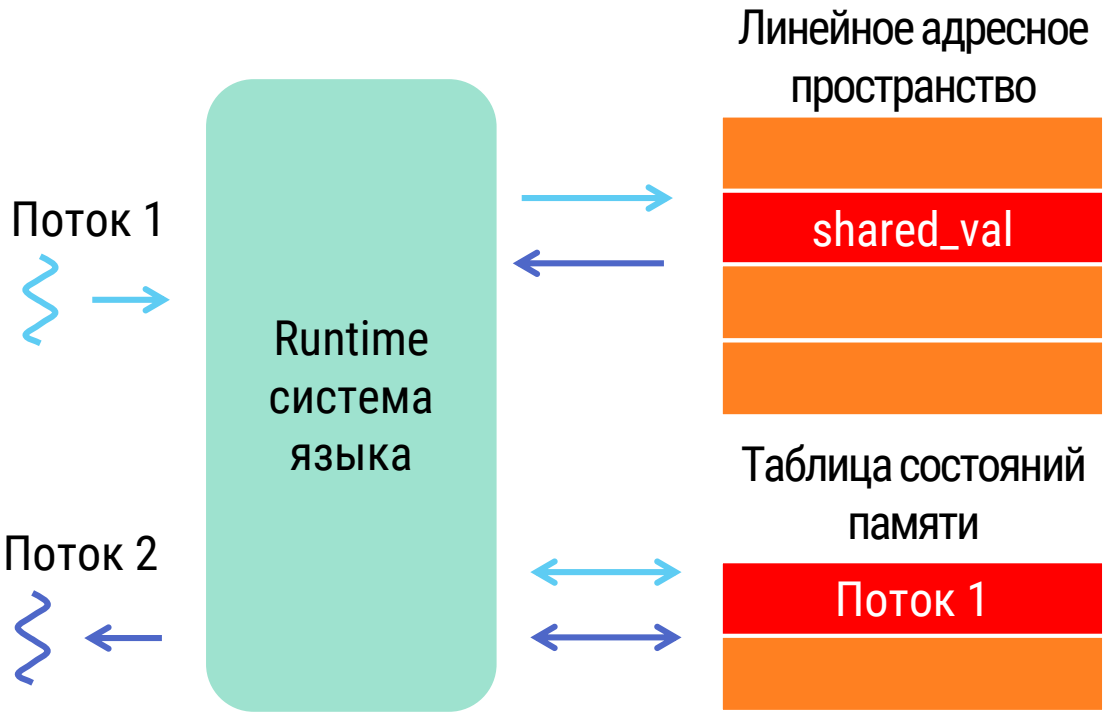
```
...
tm_begin()
  tm_write(shared_val, 5)
tm_end()
...
```

- Внутри транзакционной секции выполняется чтение глобальной переменной ***shared_val***

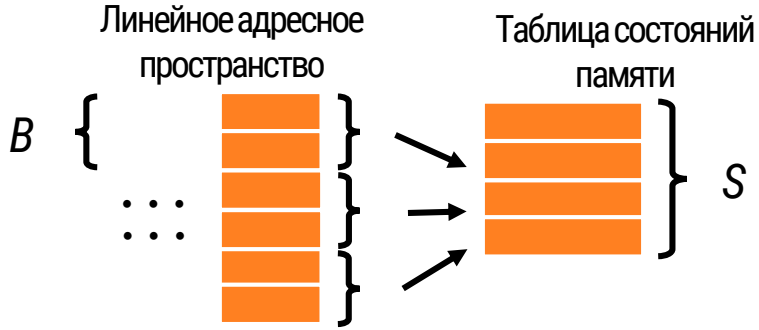
```
...
transactional_section {
    local = shared_val
}
...
```

STM-компилятор

```
...
tm_begin()
  tm_read(local, shared_val)
tm_end()
...
```



Отображение линейного адресного пространства процесса на таблицу метаданных



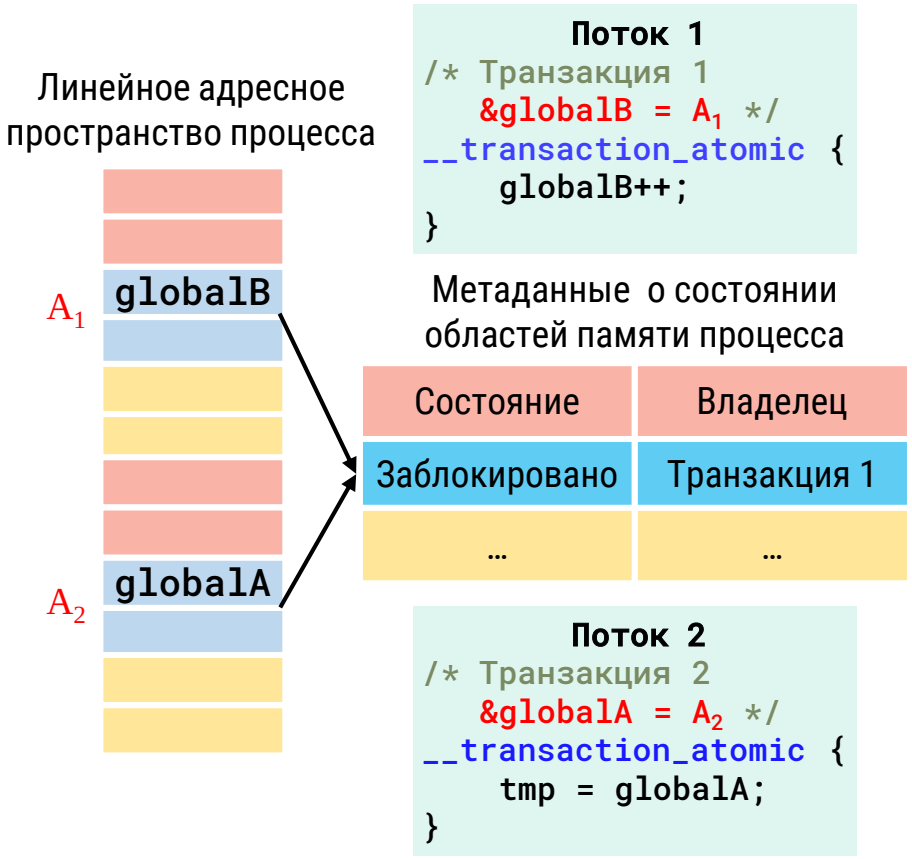
Прямое циклическое отображение адресов на таблицу фиксированного размера *S*

- Хранить информацию о состоянии области размера ***B*** байт линейного адресного пространства процесса
- Может привести к возникновению ложных конфликтов
- Используется в реализации STM в компиляторе GCC

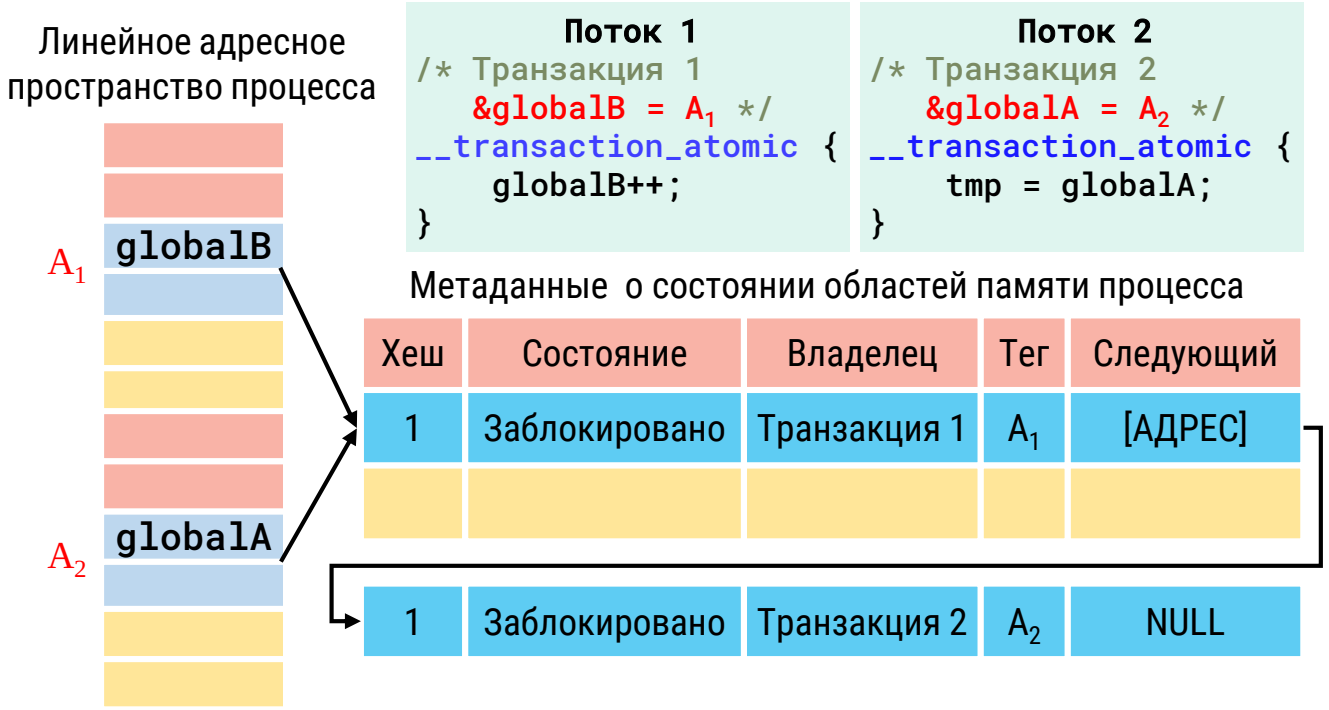
ЗАДАЧА ПРЕДОТВРАЩЕНИЯ ВОЗНИКНОВЕНИЯ ЛОЖНЫХ КОНФЛИКТОВ

Ложный конфликт – ситуация, при которой два или более потока во время выполнения транзакции обращаются к разным участкам линейного адресного пространства, отображаемым на одну и ту же строку в таблице метаданных, и как минимум один поток выполняет операцию записи

Требуется. Разработать метод предотвращения возникновения ложных конфликтов по результатам предварительного профилирования



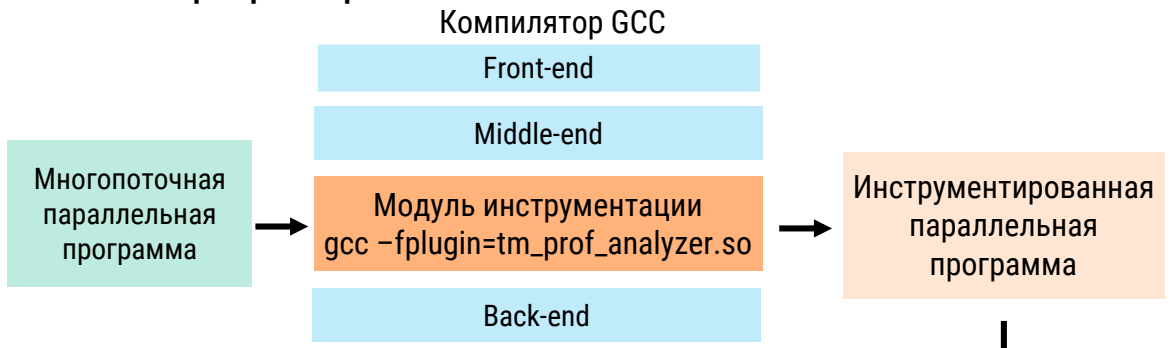
Известный метод сокращения числа ложных конфликтов на основе реорганизации таблицы метаданных [1]



[1] Zilles C., Rajwar R. **Implications of false conflict rate trends for robust software transactional memory //** URL http://zilles.cs.illinois.edu/papers/tm_false_conflicts.iiswc2007.pdf

МЕТОД СОКРАЩЕНИЯ ВОЗНИКНОВЕНИЯ ЛОЖНЫХ КОНФЛИКТОВ ПО РЕЗУЛЬТАТАМ ПРЕДВАРИТЕЛЬНОГО ПРОФИЛИРОВАНИЯ

Этап 1. Инструментация транзакционных секций для задач профилирования



Этап 2. Профилирование параллельной программы

Обработка вызова функций

Протокол выполнения параллельной программы

- Время вызова функции
- Используемые адреса памяти
- Размер используемой области памяти

Библиотека профилирования libitm_prof.so

Регистрация:
- Начала транзакций
- Операций транзакционного чтения/записи
- Отката транзакций
- Фиксации транзакций

Этап 3. Адаптивное регулирование параметров реализации runtime-системы языка

Выбор значений параметров:
B – размер блока линейного адресного пространства и *S* – размер таблицы метаданных

Пример инструментации кода транзакционной секции

Транзакционная секция

```
int a, b;
...
__transaction_atomic {
    if (a == 0)
        b = 1;
    else
        a = 0;
}
...
```

Код, сгенерированный компилятором

```
state = _ITM_beginTransaction()
<L1>:
    if (state & a_abortTransaction)
        goto <L3>;
    else
        goto <L2>;
<L2>:
    if (_ITM_LU4(&a) == 0)
        _ITM_WU4(&b, 1);
    else
        _ITM_WU4(&a, 0);
    _ITM_commitTransaction();
<L3>:
```

Инструментированная транзакционная секция

```
    state = _ITM_beginTransaction()
    tm_prof_begin(state);
<L1>:
    if (state & a_abortTransaction)
        goto <L3>;
    else
        goto <L2>;
<L2>:
    tm_prof_operation(sizeof(a));
    if (_ITM_LU4(&a) == 0) {
        tm_prof_operation(sizeof(b));
        _ITM_WU4(&b, 1);
    } else {
        tm_prof_operation(sizeof(a));
        _ITM_WU4(&a, 0);
    }
    _ITM_commitTransaction();
    tm_prof_commit();
<L3>:
```

ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ

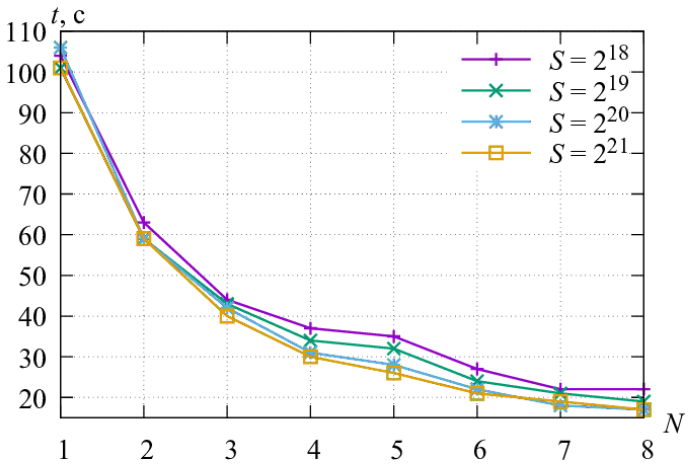
Условия запуска

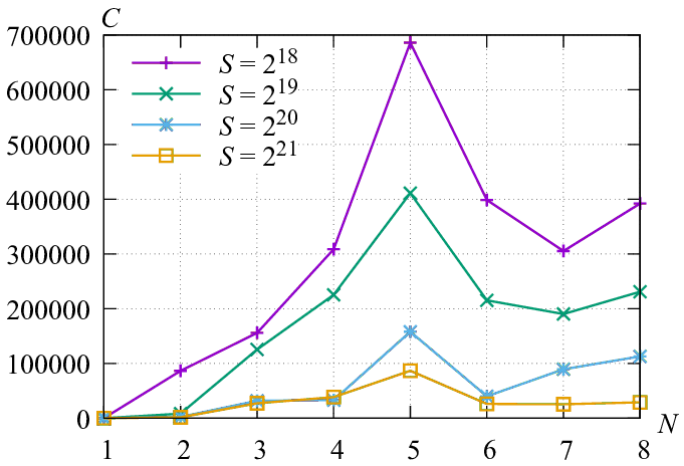
- Процессор: 2 x Intel Xeon E5420 2.5 GHz (нет Intel TSX)
- 16 GiB ОЗУ
- Число потоков варьировалось от 1 до 8
- GCC 5.1.1

Описание тестов

- Набор тестов STAMP (https://github.com/mfs409/stamp), тест реконструкции последовательности генов genome
- 5 транзакционных секций, запускаемых в цикле
- Работа со списками, хеш-таблицей, массивами
- 98% общего времени выполнения теста занимает выполнение транзакционных секций
- 99% возникающих конфликтов - ложные

Зависимость времени *t* выполнения и числа *C* ложных конфликтов от количества *N* потоков параллельной программы





Варьируемые параметры:

- *S* – размер таблицы метаданных о состоянии областей памяти
- *B* – размер блока памяти, приходящийся на одну запись таблицы метаданных

Измеряемые показатели:

- *C* – суммарное количество ложных конфликтов
- *t* – время выполнения параллельной программы

Использование предложенного метода позволяет сократить время выполнения параллельных программ в среднем на **20%**

АНАЛИЗ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ АВТОМАТИЧЕСКОЙ ВЕКТОРИЗАЦИИ

Набор тестовых циклов

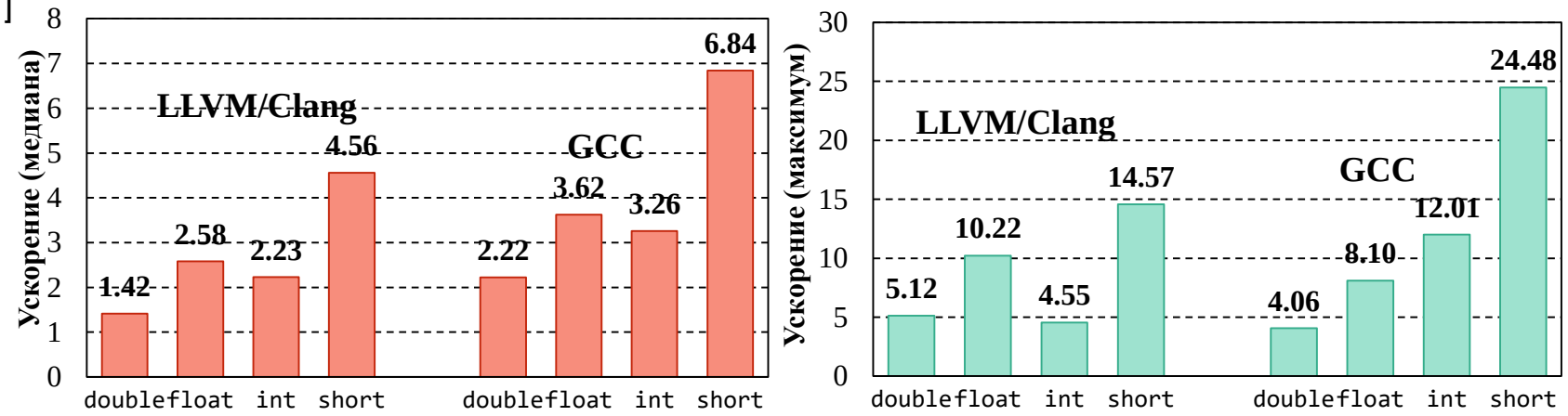
1991 **TSVC – Test Suite for Vectorizing Compilers [1]**
(122 цикла на Fortran)

Векторные ВС:
Cray, NEC, IBM, DEC, Fujitsu, Hitachi

2011 **ETSVC – Extended Test Suite for Vectorizing Compilers [2, 3]**
(151 цикл на C)

Наборы векторных инструкций:
Intel SSE/AVX, IBM AltiVec, ARM NEON SIMD, MIPS MSA

Ускорение выполнения векторизованных циклов на процессоре Intel Xeon E5-2620 v4



Категории циклов ETSVC

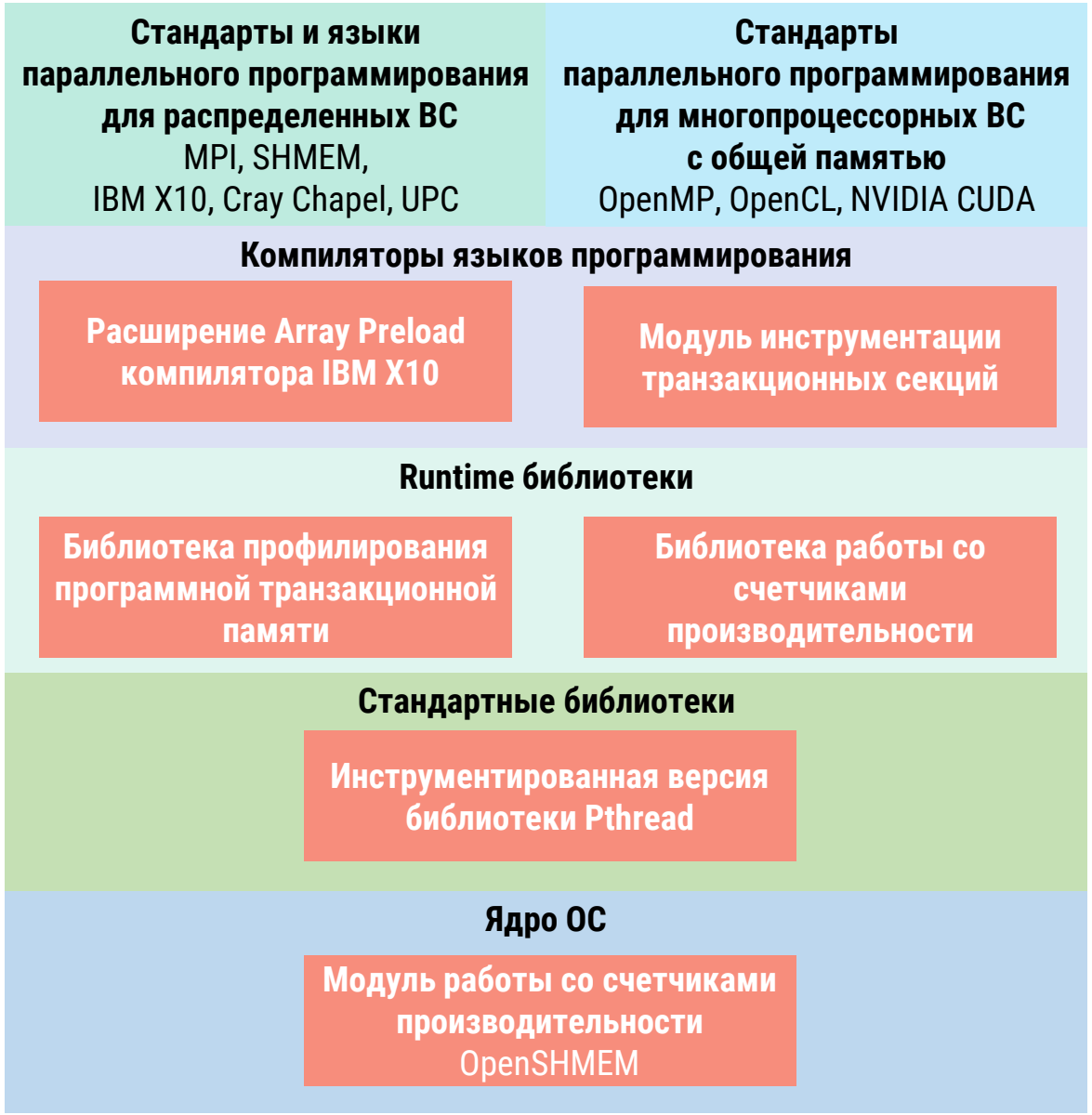
Категория	Число циклов
Анализ зависимостей по данным (dependence analysis)	36
Анализ потока управления и трансформация циклов (vectorization)	52
Распознавание идиоматических конструкций (idiom recognition)	27
Полнота понимания языка программирования (language completeness)	23
Контрольные циклы (control loops)	13

Классы невекторизованных циклов

Категория / Подкатегория	**Общее число циклов**	**Число невекторизованных циклов**
Анализ зависимостей по данным (dependence analysis)	**36**	**14**
Анализ потока управления и трансформация циклов (vectorization)	**52**	**29**
Распознавание идиоматических конструкций (idiom recognition)	**27**	**15**
Полнота понимания языка программирования (language completeness)	**23**	**6**
Контрольные циклы (control loops)	**13**	**2**

[1] Levine D., Callahan D., Dongarra J. ***A Comparative Study of Automatic Vectorizing Compilers*** // Journal of Parallel Computing. 1991. Vol. 17. pp. 1223–1244.
[2] Maleki S., Gao Ya. Garzarán M.J., Wong T., Padua D.A. ***An Evaluation of Vectorizing Compilers*** // Proc. of the Int. Conf. on Parallel Architectures and Compilation Techniques (PACT-11), 2011. pp. 372–382.
[3] ***Extended Test Suite for Vectorizing Compilers***. URL: http://polaris.cs.uiuc.edu/~maleki1/TSVC.tar.gz

ИНСТРУМЕНТАРИЙ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ



На основе созданных алгоритмов соискателем **разработаны** программные компоненты:

- Расширение Array Preload компилятора IBM X10
- Модуль инструментации транзакционных секций
- Библиотека профилирования ПТП
- Библиотека работы со счетчиками производительности
- Инструментированная версия библиотеки Pthread
- Модуль ядра ОС Linux для работы со счетчиками производительности

Предложенные пакеты вошли в состав инструментария параллельного мультипрограммирования пространственно-распределенной мультикластерной ВС, созданной членами ведущей научной школы РФ

(НШ-9505.2006.9, НШ-2121.2008.9,
НШ-5176.2010.9, НШ-2175.2012.9,
руководитель – чл.-корр. РАН В.Г. Хорошевский)

ЗАКЛЮЧЕНИЕ

В диссертации предложены архитектурно-ориентированные методы и алгоритмы организации функционирования ВС и оптимизации выполнения параллельных программ на них.

1. *Для ВС с многоуровневым параллелизмом разработаны* **средства оптимизации циклического доступа к информационным массивам** *в параллельных программах на базе модели разделенного глобального адресного пространства.*

    1. *Предложен* **алгоритм Array Preload трансформации конструкций циклической передачи потока управления подчиненным элементарным машинам**, *сокращающий время выполнения программ за счет опережающего копирования информационных массивов. В отличие от известных, разработанный метод применим к PGAS-программам, в которых на этапе компиляции неизвестно множество используемых элементов массивов.*

    2. *В моделях параллельных вычислений LogP, LogGP и Hockney построены* **оценки эффективности выполнения формируемого алгоритмом Array Preload кода**, *показывающие отсутствие функциональной зависимости времени его выполнения от количества итераций циклов.*

    3. *Выполнена* **реализация алгоритма в виде расширения компилятора языка IBM X10**. *По сравнению со стандартным алгоритмом, предложенный позволяет на кластерных ВС с сетями связи Gigabit Ethernet и InfiniBand QDR сократить время выполнения циклического доступа к элементам массивов в 1.2–2.5 раза.*

ЗАКЛЮЧЕНИЕ (ПРОДОЛЖЕНИЕ)

2. *Для многопроцессорных вычислительных узлов с общей памятью разработаны и исследованы* **методы оптимизации выполнения многопоточных программ.**

    1. *Предложен* **метод сокращения числа ложных конфликтов в многопоточных программах на базе программной транзакционной памяти**. *В основе метода лежит подбор (суб)оптимальных параметров таблиц обнаружения конфликтов в реализации транзакционной памяти по результатам предварительного профилирования целевой программы.*

    2. *Выполнена* **программная реализация метода сокращения числа ложных конфликтов в расширении компилятора GCC**. *Использование предложенного метода позволяет сократить время выполнения параллельных программ в среднем на 20%, что экспериментально показано на тестовых программах из пакета STAMP.*

    3. *Для известных алгоритмов автоматической векторизации циклов в открытых компиляторах GCC и LLVM/Clang* **выявлены классы трудно векторизуемых циклов** *из тестового набора ETSVC.* **Установлено, что на архитектуре Intel 64 известные алгоритмы способны векторизовать от 34% до 52% циклов пакета ETSVC.** *Построенное подмножество циклов составляет базисный набор для анализа эффективности ядер автовекторизаторов оптимизирующих компиляторов для векторных процессоров класса «регистр-регистр».*

    4. *Создан* **инструментарий анализа эффективности использования микроархитектурных возможностей ядер суперскалярных процессоров ВС.** *В отличие от известных пакетов, предложенные программные средства позволяют анализировать загрузку суперскалярного конвейера архитектуры Intel 64 потоком инструкций с точностью до нескольких команд ассемблера.*

ЗАКЛЮЧЕНИЕ (ПРОДОЛЖЕНИЕ)

3. *Выполнено* **развитие программно-аппаратной конфигурации мультикластерной ВС.** *В состав системы введены сегменты на базе вычислительных узлов с сопроцессорами Intel Xeon Phi и графическими процессорами NVIDIA. Программный инструментарий системы расширен разработанными автором пакетами оптимизации использования многоуровневого параллелизма ВС в параллельных программах.*

ПУБЛИКАЦИИ И АПРОБАЦИЯ РЕЗУЛЬТАТОВ

По теме диссертации автором опубликовано 23 работы:

- в журналах из перечня ВАК РФ: 4

- в изданиях, индексируемых Scopus и Web of Science: 2

- свидетельства о государственной регистрации программ для ЭВМ: 3

Апробация результатов. *Основные результаты работы докладывались и обсуждались на международных, всероссийских и региональных научных конференциях, в их числе:*

международные конференции: *«Parallel Computing Technologies» (PaCT-2015, Petrozavodsk), «Параллельные вычислительные технологии» (ПаВТ-2016, г. Архангельск), «Суперкомпьютерные технологии: разработка, программирование, применение» (СКТ-2014, СКТ-2016, г. Геленджик), «Открытая конференция по компиляторным технологиям» (2015, г. Москва);*

российские конференции: *«Актуальные проблемы вычислительной и прикладной математики» (АПВПМ-2015, г. Новосибирск), «Новые информационные технологии в исследовании сложных структур» (ICAM-2014, г. Томск), Сибирская конференция по параллельным и высокопроизводительным вычислениям (2015, г. Томск).*

Основные этапы исследования выполнены в ходе выполнения работ по проектам Российского фонда фундаментальных исследований №№ 15-07- 00653, 15-37- 20113; Совета Президента РФ по поддержке ведущих научных школ № НШ-2175.2012.9 (руководитель – чл.-корр. РАН В.Г. Хорошевский); грантов Новосибирской области для молодых ученых, стипендии Президента РФ для аспирантов.

ПРАКТИЧЕСКАЯ ЗНАЧИМОСТЬ РАБОТЫ, ВНЕДРЕНИЯ

Методы и алгоритмы доведены до практической реализации в виде программных пакетов (получено 3 авторских свидетельства о гос. регистрации программ для ЭВМ)

Внедрение в Центре параллельных вычислительных технологий СибГУТИ – системное программное обеспечение анализа и оптимизации использования многоуровневого параллелизма ВС в параллельных программах

Результаты работы внедрены в учебный процесс СибГУТИ, в систему параллельного мультипрограммирования пространственно-распределенной ВС

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Спасибо за внимание!

Кулагин Иван Иванович

Диссертация на соискание ученой степени кандидата технических наук
Специальность: 05.13.15 «Вычислительные машины, комплексы и компьютерные сети»

Научный руководитель –
доктор технических наук, доцент,
Курносов Михаил Георгиевич

г. Новосибирск, 15 февраля 2018 г.

ПУБЛИКАЦИИ

Статьи в журналах из перечня ВАК РФ

1. **Кулагин, И.И.** *О спекулятивном выполнении критических секций на вычислительных системах с общей памятью / И.И. Кулагин, М.Г. Курносов // Известия Южного федерального университета. Технические науки. – 2016. – No 11. – С. 54-64.*
2. **Кулагин, И.И.** *Оптимизация обнаружения конфликтов в параллельных программах с транзакционной памятью / И.И. Кулагин, М.Г. Курносов // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. – 2016. – Т. 5, No 4. – С. 46-60.*
3. **Кулагин, И.И.** *Инструментация и оптимизация выполнения транзакционных секций многопоточных программ / И.И. Кулагин, М.Г. Курносов // Труды Института системного программирования РАН. – 2015. – Том 27. Выпуск 6. – С. 135-150.*
4. **Кулагин, И.И.** *Эвристические алгоритмы оптимизации информационных обменов в параллельных PGAS-программах / И.И. Кулагин, А.А. Пазников, М.Г. Курносов // Вестник СибГУТИ. – No 3. – 2014. – С. 52-66.*

Публикации в изданиях, индексируемых Scopus и Web of Science

5. **Kulagin, I.** *Optimization of Conflict Detection in Parallel Programs with Transactional Memory / I. Kulagin, M. Kurnosov // Proc. of 10th Annual International Scientific Conference on Parallel Computing Technologies (PCT-2016). – CEUR-WS, 2016 – Vol. 1576. – P. 582-594.*
6. **Kulagin, I.** *Heuristic Algorithms for Optimizing Communications in Parallel PGAS-programs / I. Kulagin, A. Paznikov, M. Kurnosov // Proc. of the 13th International Conference on Parallel Computing Technologies (PaCT-2015). – Springer LNCS, 2015. – Vol. 9251. – P. 405-409.*

Свидетельства о государственной регистрации программ для ЭВМ

7. *Свидетельство 2017660065 РФ. Программа детального анализа производительности выполнения кода на архитектуре Intel 64 : свидетельство об официальной регистрации программы для ЭВМ / **И.И. Кулагин**, М.Г. Курносов; заявитель и патентообладатель СибГУТИ; заявл. 17.07.2017, опубл. 14.09.2017.*
8. *Свидетельство 2016660098 РФ. Программа оптимизации выполнения транзакционных секций в параллельных программах для вычислительных систем с общей памятью : свидетельство об официальной регистрации программы для ЭВМ / **И.И. Кулагин**, М.Г. Курносов; заявитель и патентообладатель СибГУТИ; заявл. 25.07.2016, опубл. 06.09.2016.*
9. *Свидетельство 2015619554 РФ. Программа компиляторной оптимизации доступа к удаленным массивам в программах на языке IBM X10: свидетельство о государственной регистрации программы для ЭВМ / **И.И. Кулагин**, М.Г. Курносов; заявитель и патентообладатель СибГУТИ; заявл. 14.07.2015, опубл. 08.09.2015.*

ПУБЛИКАЦИИ

Публикации в сборниках трудов и материалах конференций

10. *Молдованова, О.В. Векторизация циклов в открытых компиляторах для архитектур с короткими векторными регистрами / О.В. Молдованова, **И.И. Кулагин**, М.Г. Курносов // Сборник трудов тринадцатой международной Азиатской школа-семинар «Проблемы оптимизации сложных систем» в рамках международной мультиконференции IEEE SIBIRCON-2017. – 2017. – С. 70-78.*
11. ***Кулагин, И.И.** Оптимизация обнаружения конфликтов в параллельных программах с транзакционной памятью / И.И. Кулагин, М.Г. Курносов // Труды международной научной конференции «Параллельные вычислительные технологии (ПаВТ)-2016», 2016. – С. 4.*
12. ***Кулагин, И.И.** Подход к сокращению ложных конфликтов в параллельных программах на базе транзакционной памяти / И.И. Кулагин, М.Г. Курносов // Тезисы докладов Сибирской конференции по параллельным и высокопроизводительным вычислениям (СКПВВ-2015), Томск, 2015.– С. 48.*
13. ***Кулагин, И.И.** Подход к архитектурно-ориентированной оптимизации программ для архитектуры Intel 64 // Материалы Российской научно-технической конференции «Обработка информации и математическое моделирование», 2017 – Новосибирск: СибГУТИ, 2017. – С. 300-310.*
14. ***Кулагин, И.И.** О подходах к реализации программной транзакционной памяти / И.И. Кулагин, М.Г. Курносов // Материалы Российской научно-технической конференции «Обработка информации и математическое моделирование», 2016. – Новосибирск: СибГУТИ, 2016. – С. 331-338.*
15. ***Кулагин, И.И.** Анализ обнаружения ложных конфликтов в приложениях с программной транзакционной памятью / И.И. Кулагин, М.Г. Курносов // Материалы Российской научно-технической конференции «Обработка информации и математическое моделирование», 2015. – С. 335-337.*
16. ***Кулагин, И.И.** Методы оптимизации передачи массивов в параллельных программах на языке IBM X10 / И.И. Кулагин, М.Г. Курносов // Материалы докладов 10-ой Российской конференции с международным участием «Новые информационные технологии в исследование сложных структур» (ICAM-2014). – Томск: НТЛ, 2014. – С. 5-6.*
17. ***Кулагин, И.И.** О спекулятивном выполнении критических секций на вычислительных системах с общей памятью / И.И. Кулагин, М.Г. Курносов // Материалы Всероссийской научно-технической конференции «Суперкомпьютерные технологии» (СКТ-2016), 2016. – Т. 1. – С. 170-174.*
18. ***Кулагин, И.И.** Повышение эффективности циклического доступа к удаленным массивам в программах на языке IBM X10 // Сборник статей Всероссийской научно-практической конференции: Многоядерные процессоры, параллельное программирование, ПЛИС, системы обработки сигналов. – Изд-во Алт. ун-та, 2015. – С. 129-136.*
19. ***Кулагин, И.И.** Алгоритмы оптимизации ложных конфликтов в параллельных программах на базе транзакционной памяти / И.И. Кулагин, М.Г. Курносов // Материалы международной конференции «Актуальные проблемы вычислительной и прикладной математики» (АПВПМ-2015, AMCA-2015), Новосибирск, 2015. – С. 416-422.*

ПУБЛИКАЦИИ

Публикации в сборниках трудов и материалах конференций (продолжение)

20. **Кулагин, И.И.** *Оптимизация доступа к удаленным массивам в программах на языке IBM X10 // Материалы 52-й Международной научной студенческой конференции МНСК-2014: Информационные технологии / Новосиб. гос. ун-т. – Новосибирск, 2014. С. 168.*
21. **Кулагин, И.И.** *Исследование эффективности доступа к распределенным массивам в программах на языке IBM X10 / И.И. Кулагин, М.Г. Курносов // Материалы Российской научно-технической конференции «Обработка информационных сигналов и математическое моделирование». – Новосибирск, 2014. – С. 77-79.*
22. **Кулагин, И.И.** *Оптимизация информационных обменов в параллельных PGAS-программах / И.И. Кулагин, А.А. Пазников, М.Г. Курносов // Материалы 3-й Всероссийской научно-технической конференции «Суперкомпьютерные технологии» (СКТ-2014), 2014. – Т.1 – С. 158-162.*
23. **Кулагин, И.И.** *Оптимизация выполнения MPI-программ по результатам их профилирования / М.Г. Курносов, И.И. Кулагин // Материалы Российской научно-технической конференции «Обработка информационных сигналов и математическое моделирование». Новосибирск, 2012. – С. 159-160.*

Специальность: 05.13.15 Вычислительные машины, комплексы и компьютерные сети

Формула специальности:

Вычислительные машины, комплексы и компьютерные сети – специальность, включающая исследования и разработку научных основ архитектурных, структурных, логических и технических принципов создания вычислительных машин, комплексов и компьютерных сетей, организации арифметической, логической, символьной и специальной обработки данных, хранения и ввода-вывода информации, параллельной и распределенной обработки информации, многопроцессорных и многомашинных вычислительных систем, сетевых протоколов и служб передачи данных в компьютерных сетях, взаимодействия и защиты компьютерных сетей. Важное народно-хозяйственное значение данной специальности состоит в создании и совершенствовании теоретической и технической базы вычислительных машин, комплексов и компьютерных сетей, обладающих высокими качественными и эксплуатационными показателями и обеспечивающих ускорение научно-технического прогресса.

Области исследований:

1. Разработка научных основ создания вычислительных машин, комплексов и компьютерных сетей, исследования общих свойств и принципов функционирования вычислительных машин, комплексов и компьютерных сетей.

2. Теоретический анализ и экспериментальное исследование функционирования вычислительных машин, комплексов и компьютерных сетей с целью улучшения их технико-экономических и эксплуатационных характеристик.

3. Разработка научных методов и алгоритмов организации арифметической, логической, символьной и специальной обработки данных, хранения и ввода-вывода информации.

4. Разработка научных методов и алгоритмов организации параллельной и распределенной обработки информации, многопроцессорных, многомашинных и специальных вычислительных систем.

5. Разработка научных методов и алгоритмов создания структур и топологий компьютерных сетей, сетевых протоколов и служб передачи данных в компьютерных сетях, взаимодействия компьютерных сетей, построенных с использованием различных телекоммуникационных технологий, мобильных и специальных компьютерных сетей, защиты компьютерных сетей и приложений.

6. Разработка научных методов, алгоритмов и программ, обеспечивающих надежность, контроль и диагностику функционирования вычислительных машин, комплексов и компьютерных сетей.

Примечание:

Специальность не включает исследования и разработки в областях:
элементы и устройства вычислительной техники и систем управления;
автоматизированные системы управления технологическими процессами и производством;
математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей;
телекоммуникационные системы. Эти области исследования и разработки включены в специальности: 05.13.05, 05.13.06, 05.13.11, 05.12.13.

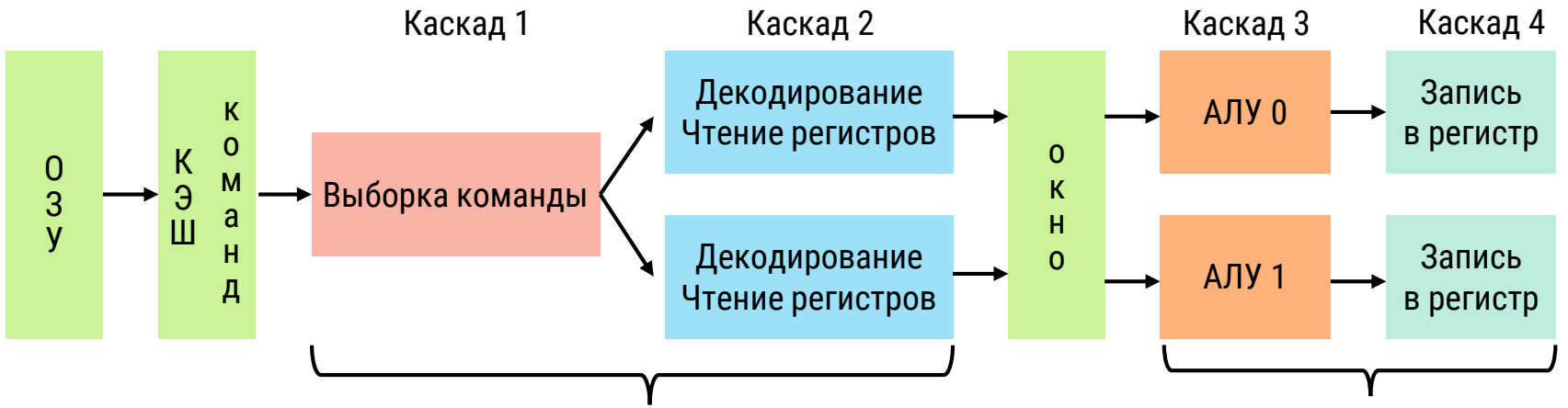
РЕКОМЕНДАЦИИ И ПЕРСПЕКТИВЫ ДАЛЬНЕЙШЕЙ РАЗРАБОТКИ ТЕМЫ

1. *Разработка методов организации отказоустойчивого (живучего) мультипрограммного функционирования большемасштабных мультиархитектурных ВС при выполнении PGAS-программ.*

2. *Развитие оценочных моделей для расстановки приоритетов (полу)автоматического совместного использования различных форм параллелизма в программах: передача сообщений, многопоточность, векторизация кода.*

АКТУАЛЬНОСТЬ ТЕМЫ ИССЛЕДОВАНИЯ

Уровень суперскалярного ядра процессора



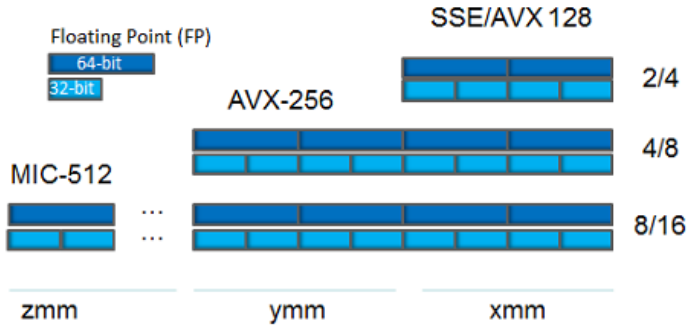
В лучшем случае,
максимальное ускорение от ILP – *n* раз,
n – количество АЛУ
ILP [1, 2] – Instruction Level Parallelism/
Параллелизм уровня команд

Например:
Микроархитектура Skylake: 14 каскадов, 8 АЛУ
Максимальное ускорение – 8 раз

Причины простоя АЛУ:
- Конфликты конвейера
- Зависимость по данным между командами
- Конфликты по управлению (условные переходы)
- Нехватка регистров для размещения операндов
- Особенности работы с кэш-памятью
- И др.

Уровень векторных АЛУ[3]

Векторные регистры (Intel 64, Intel Xeon Phi)



Наборы векторных команд:

- Intel MMX/SSE/AVX/AVX-3, AVX-512
- IBM AltiVec
- ARM NEON SIMD
- MIPS MSA

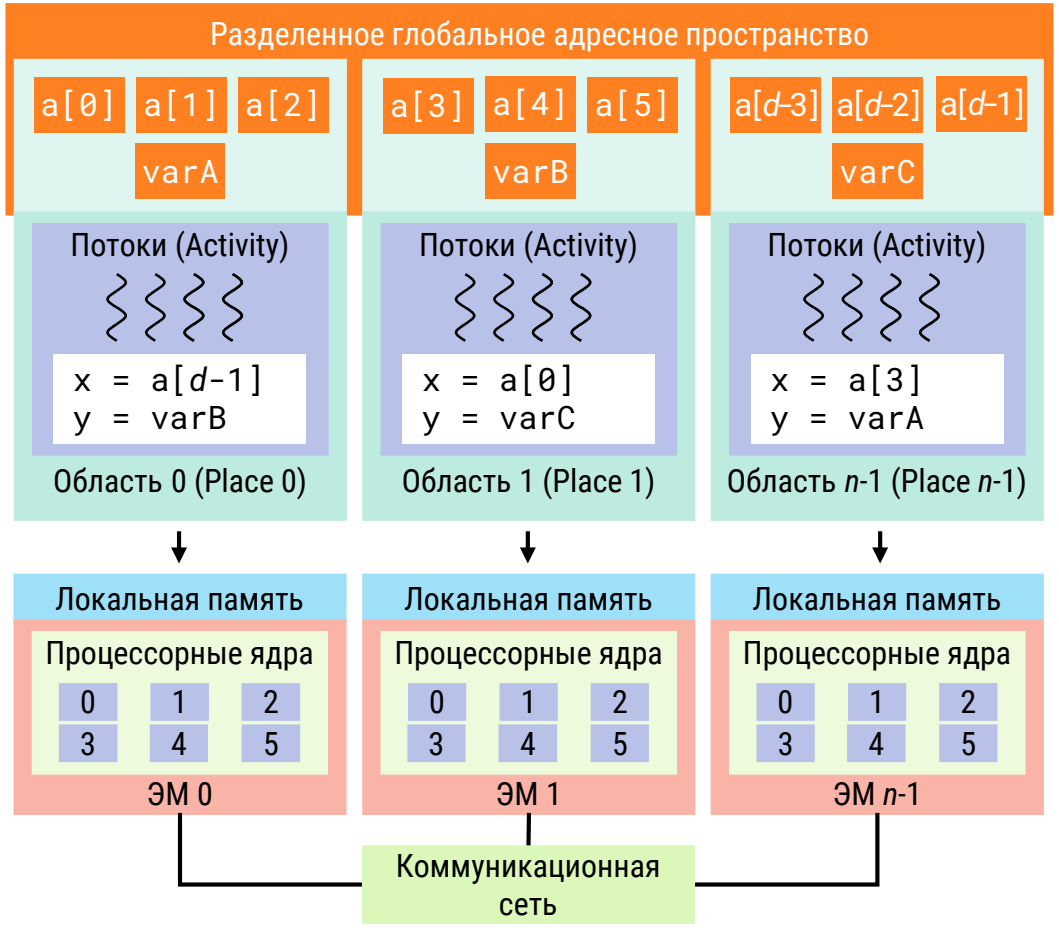
Достигаемые ускорения:

Тип данных	Intel AVX-512 (регистры 512 бит)	ARMv8 Scalable Vector Extension (регистры 128-2048 бит, RIKEN Post-K supercomputer, 2020)
double	x8	**X32**
float	x16	**x64**
int	x16	x64
short int	x32	x128

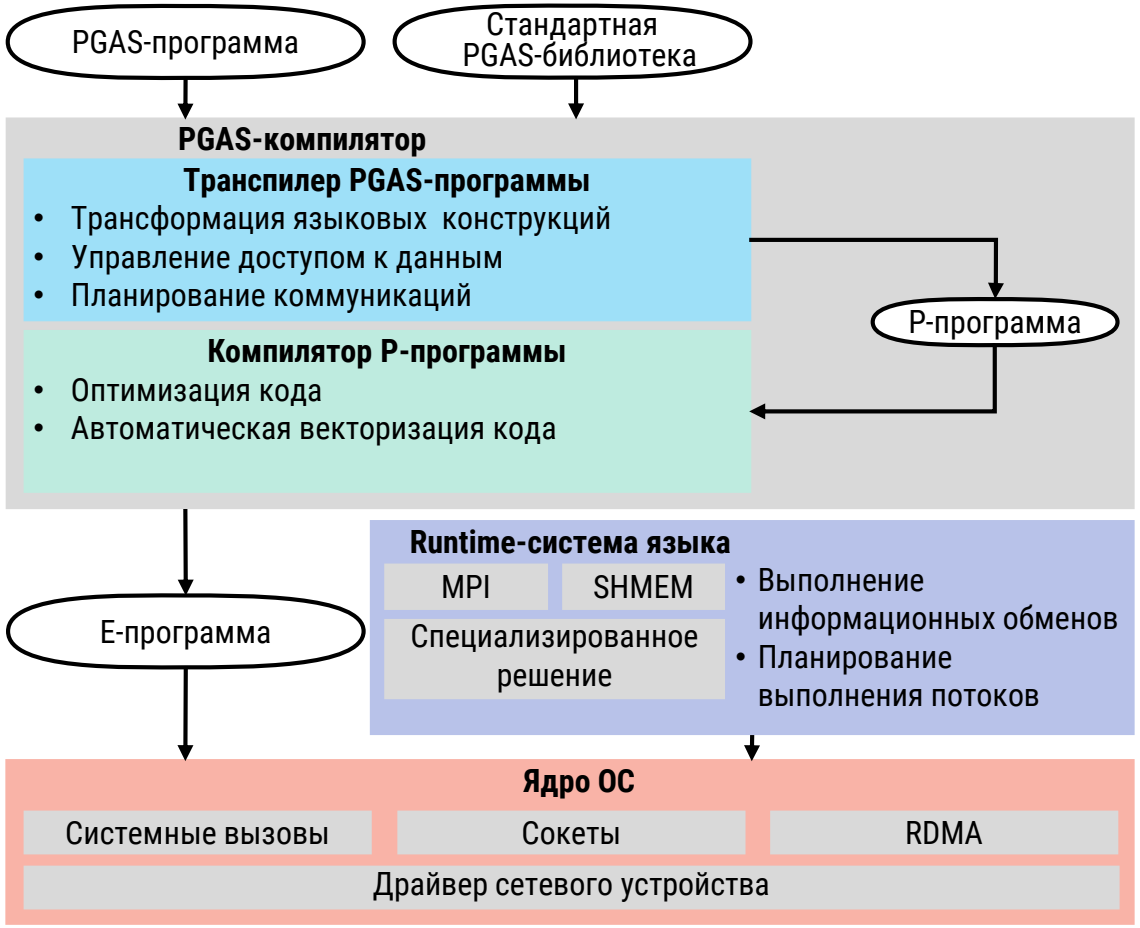
[1] S.-M. Moon, K. Ebcioglu. **Parallelizing Nonnumerical Code with Selective Scheduling and Software Pipelining.** ACM TOPLAS, Vol 19, No. 6, pages 853-898, November 1997.
[2] J. Llosa, A. Gonzalez, E. Ayguade, and M. Valero. **Swing modulo scheduling: A lifetime sensitive approach.** In Proceedings of the 1996 Conference on Parallel Architectures and Compilation Techniques (PACT '96), pages 80–87, Boston, Massachusetts, USA, October 1996.
[3] **Векторизация программ: теория, методы, реализация.** Сб. статей: Пер. с англ. и нем. М.: Мир, 1991. 275 с.

АКТУАЛЬНОСТЬ ТЕМЫ ИССЛЕДОВАНИЯ

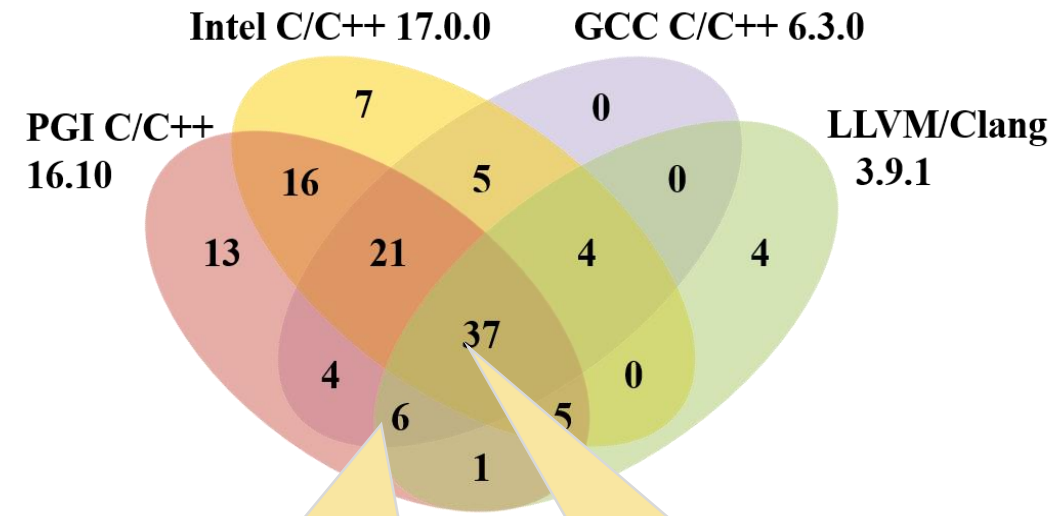
Модель
разделенного глобального адресного пространства – PGAS



Стек программного обеспечения PGAS

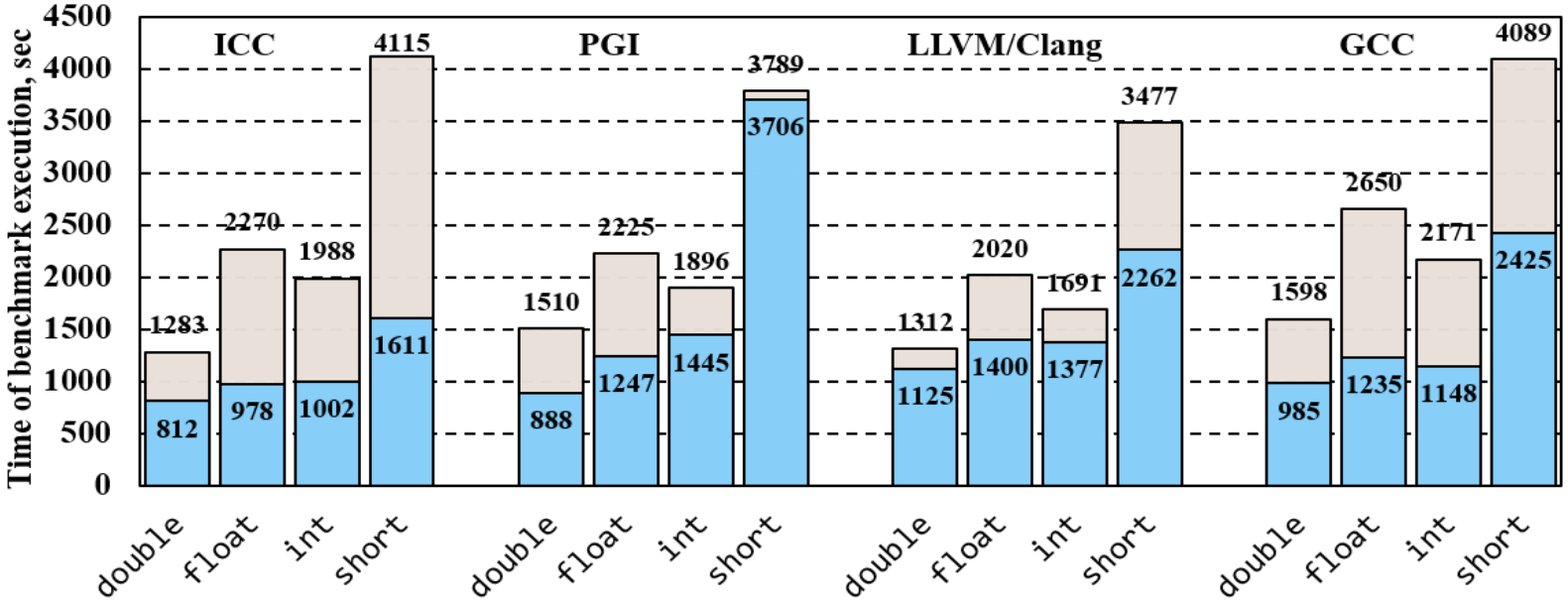


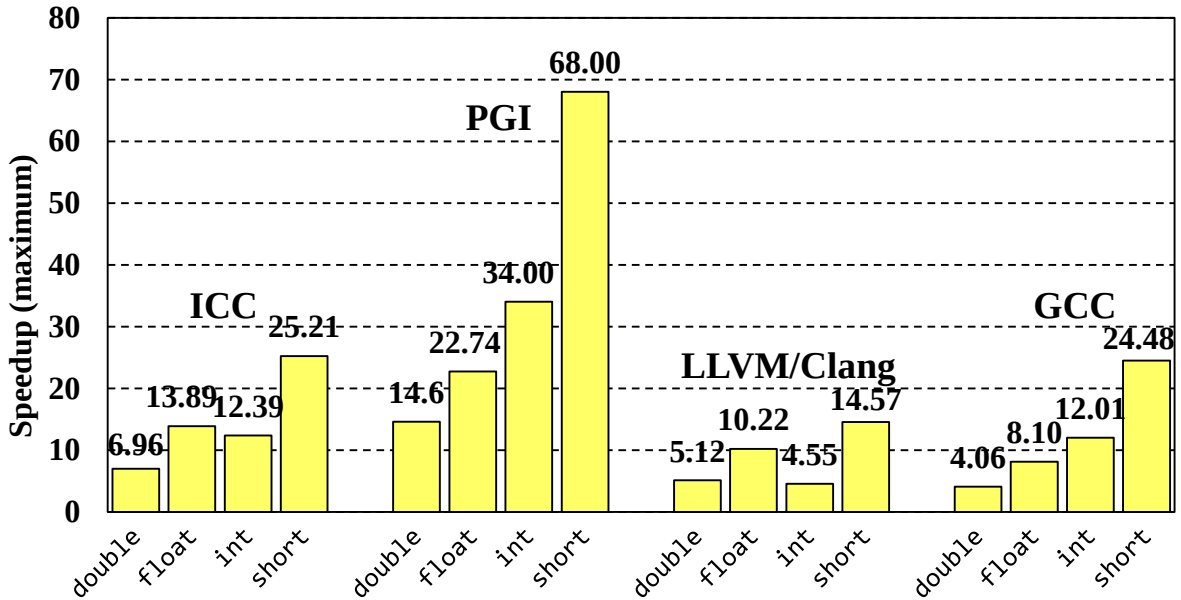
КЛАССЫ НЕВЕКТОРИЗОВАННЫХ ЦИКЛОВ

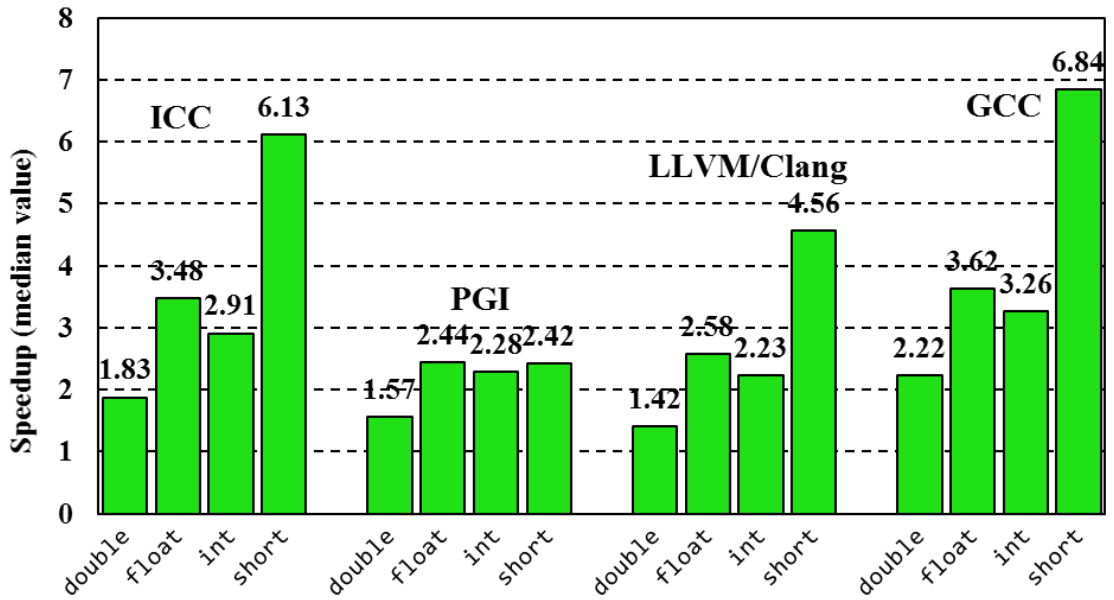


Категории циклов / подкатегории	Общее кол-во циклов	Невектор. циклы
Dependence Analysis	**36**	**9**
Linear Dependence	14	2
Induction Variable Recognition	8	3
Nonlinear Dependence	1	1
Control Flow	3	1
Symbolics	6	2
Vectorization	**52**	**11**
Loop Distribution	3	2
Loop Interchange	6	2
Node Splitting	6	4
Scalar and Array Expansion	12	2
Control Flow	14	1
Idiom Recognition	**27**	**6**
Recurrences	3	3
Search Loops	2	1
Loop Rerolling	4	1
Reductions	15	1
Language Completeness	**23**	**2**
Nonlocal GOTO	2	2

РЕЗУЛЬТАТЫ ВЕКТОРИЗАЦИИ (INTEL XEON E5-2620 v4)







РЕЗУЛЬТАТЫ ВЕКТОРИЗАЦИИ (INTEL XEON PHI 3120A)