Московский физико-технический институт (государственный университет)
Факультет радиотехники и кибернетики
Кафедра информатики и вычислительной техники

Программная конвейеризация циклов в оптимизирующем компиляторе

Выпускная квалификационная работа
(магистерская диссертация)

Выполнил:

студент 918 группы Земляков Дмитрий Александрович

Научный руководитель:

к.т.н. Ермолицкий Александр Викторович

Москва 2015

Проблематика

- Программная конвейеризация циклов является важной оптимизацией для полного использования производительности вычислительных систем.

- Два основных метода конвейеризации планируют цикл с использованием сложного итеративного эвристического алгоритма или с распознаванием устойчивого состояния в раскрученном цикле.

Пример программной конвейеризации (modulo scheduling)

Типичный универсальный цикл:

```
for(i = 0; i < n; i++)
    d[i] = a[i] * b[i] + c;
```

План итерации:

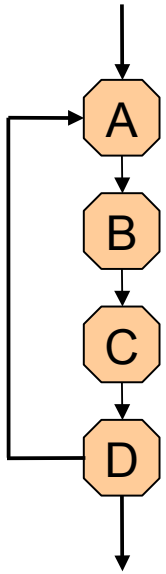
```
1 ld  Vs0 Vs2 → Vs3
2 ld  Vs1 Vs2 → Vs4
3 mul Vs3 Vs4 → Vs5
4
5 add Vs5 Vs8 → Vs6
6
7 st  Vs9 Vs2 Vs6
```

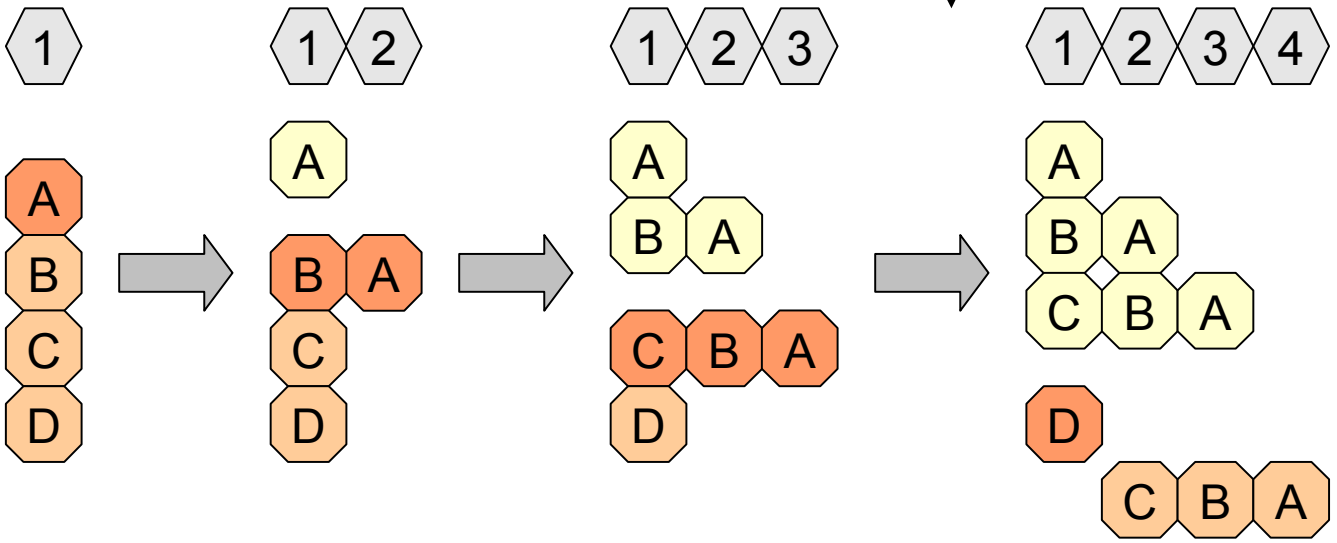
Программная конвейеризация:

```
 1    ld
 2    ld
 3    mul    ld
 4           ld
 5           mul    ld
 6    add           ld
 7                  mul    ld
 8    st     add           ld
 9                        mul
10           st     add
11
12                  st     add
13
14                         st
```

Пример программной конвейеризации (Enhanced Pipeline Scheduling)

```
for(i = 0; i < n; i++)
   a[i] = b[i] + c;
   d[i] = a[i] + e[i]
   f[i] = g[i-1] + d[i]
   g[i] = f[i] + j[i]
```





Постановка задачи

1. Реализовать оптимизацию программной конвейеризации циклов на основе алгоритма EPS
2. Внедрить оптимизацию в промышленный оптимизирующий компилятор для архитектуры SPARC
3. Адаптировать оптимизацию для архитектуры «Эльбрус»
4. Провести замеры эффективности оптимизации на пакетах тестов производительности SPEC

Реализация оптимизации для решения поставленной задачи

Основные этапы:

- Анализ необходимости и возможности применения
- Динамическая проверка числа итераций
- Анализ эффективности применения
- Перестроение цикла с коррекцией всех необходимых аналитических структур

Анализ циклов процедуры на возможность конвейеризации

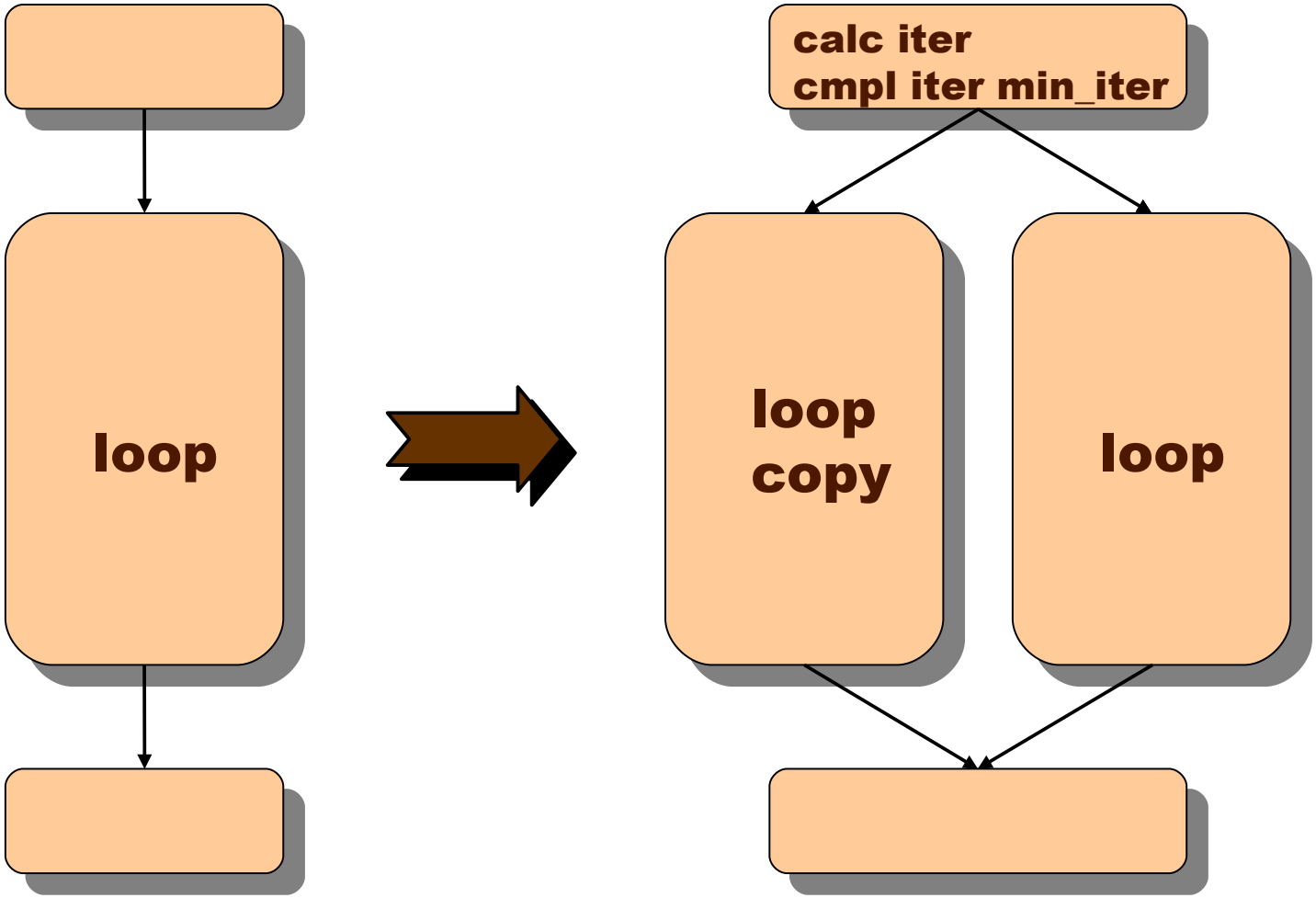
1) Проверка структуры:
   - 1 вход
   - 1 выход
   - 1 обратная дуга
2) Проверка операций:
   - min < количество < max
   - нет ассемблерных вставок
3) Проверка итераций:
   - LDA-информация
   - директива: #pragma loop count
   - профильная информация

Динамическая проверка

При невозможности статически определить количество итераций цикла выполняется динамическая проверка:

- Анализ возможности
  - Поиск базовой индуктивности
  - Будет ли эффект от копирования
- Перестроение цикла
  - Копирование исходного цикла
  - Создание ветвления управления в прологе

Динамическая проверка



Конвейеризация цикла

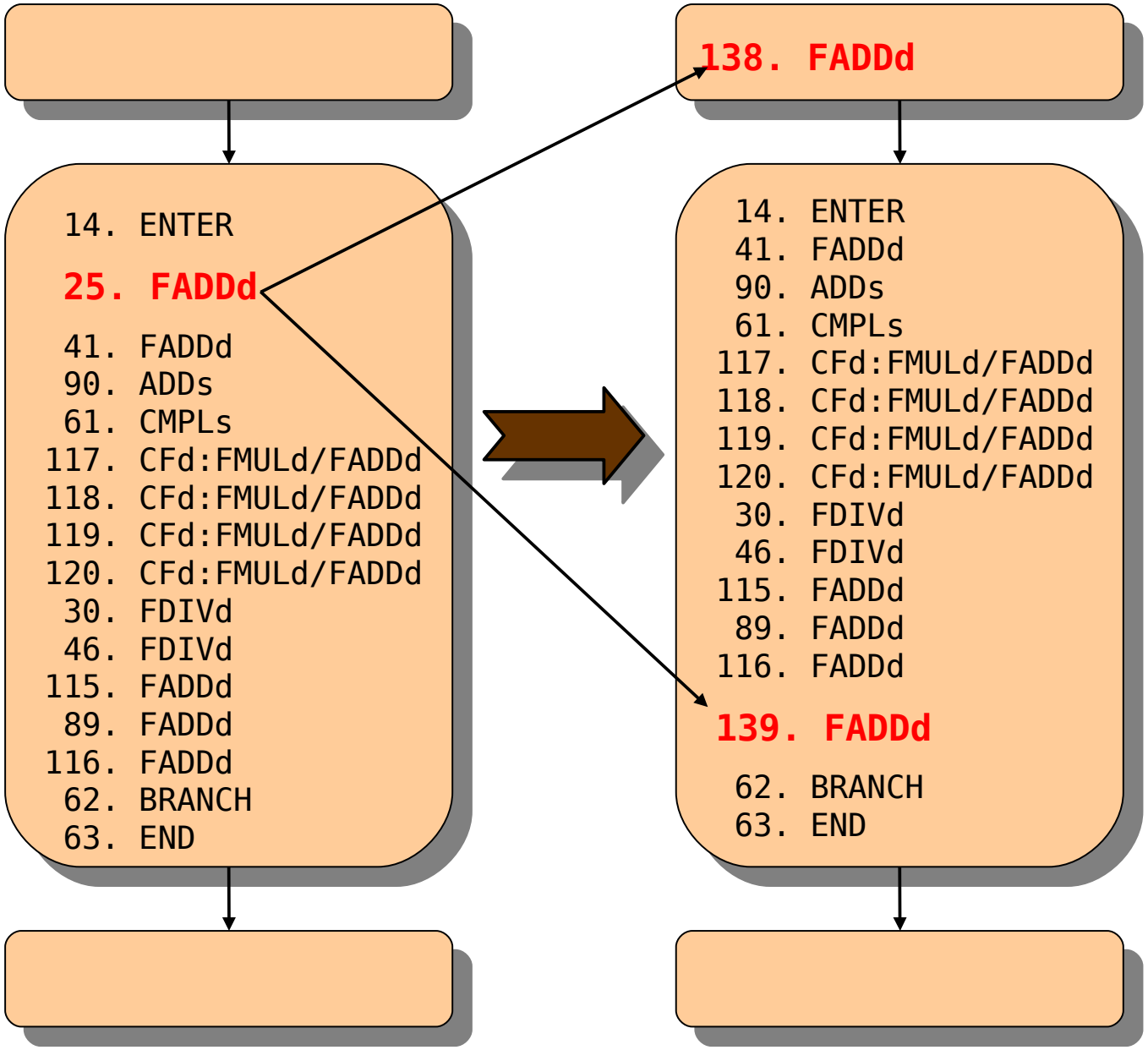
- Определение длины критического пути
- Ресурсное планирование
- Выбор операций рабочего множества
- Проверка эффективности переноса
- Перенос операций на итерацию вверх

Пример работы оптимизации

```
for (i=0; i<N; i++) {
        s += x / (x + a0 * (x + b0 * (x + c0))) +
             x / (x + a1 * (x + b1 * (x + c1)));
        x += 1.0;
    }
```

```
        14. ENTER
 1      25. FADDd          Fd70      Fd64      ->  Fd22
 4      41. FADDd          Fd70      Fd44      ->  Fd48
        90. ADDs           Vs0       0x1       ->  Vs0
        61. CMPLs          Vs0       Vs13      ->  P0
 2     117. CFd:FMULd/FADDd    Fd0   Fd22      Fd70  ->  Fd26
 5     118. CFd:FMULd/FADDd    Fd4   Fd48      Fd70  ->  Fd52
 3     119. CFd:FMULd/FADDd    Fd8   Fd26      Fd70  ->  Fd30
 6     120. CFd:FMULd/FADDd    Fd68  Fd52      Fd70  ->  Fd56
 7      30. FDIVd          Fd70      Fd30      ->  Fd32
        46. FDIVd          Fd70      Fd56      ->  Fd58
       115. FADDd          Fd2       Fd32      ->  Fd72
        89. FADDd          Fd70      Fd66      ->  Fd70
       116. FADDd          Fd72      Fd58      ->  Fd2
        62. BRANCH  .c     [14]
        63. END
```

Пример работы оптимизации



Пример работы оптимизации

```
T=18  0  90. ADDs
T=19  0  61. CMPLs
T=20  0 117. CFd:FMULd/FADDd
T=21  0  41. FADDd
T=25  0 118. CFd:FMULd/FADDd
T=28  0 119. CFd:FMULd/FADDd
T=33  0 120. CFd:FMULd/FADDd
T=36  0  30. FDIVd
T=53  0  46. FDIVd
T=54  0 115. FADDd
T=55  0  89. FADDd
T=70  0 116. FADDd
T=71  1  62. BRANCH
T=71  0 147. FADDd
```

```
T=30  1 276. ADDs
T=32  0 278. CMPLs
T=33  0 115. FADDd
T=34  0  46. FDIVd
T=35  0  89. FADDd
T=39  0 139. FADDd
T=40  0 196. FADDd
T=43  0 158. CFd:FMULd/FADDd
T=44  0 235. CFd:FMULd/FADDd
T=51  0 177. CFd:FMULd/FADDd
T=52  0 254. CFd:FMULd/FADDd
T=59  0 216. FDIVd
T=60  0 116. FADDd
T=60  1  62. BRANCH
```

53 → 30

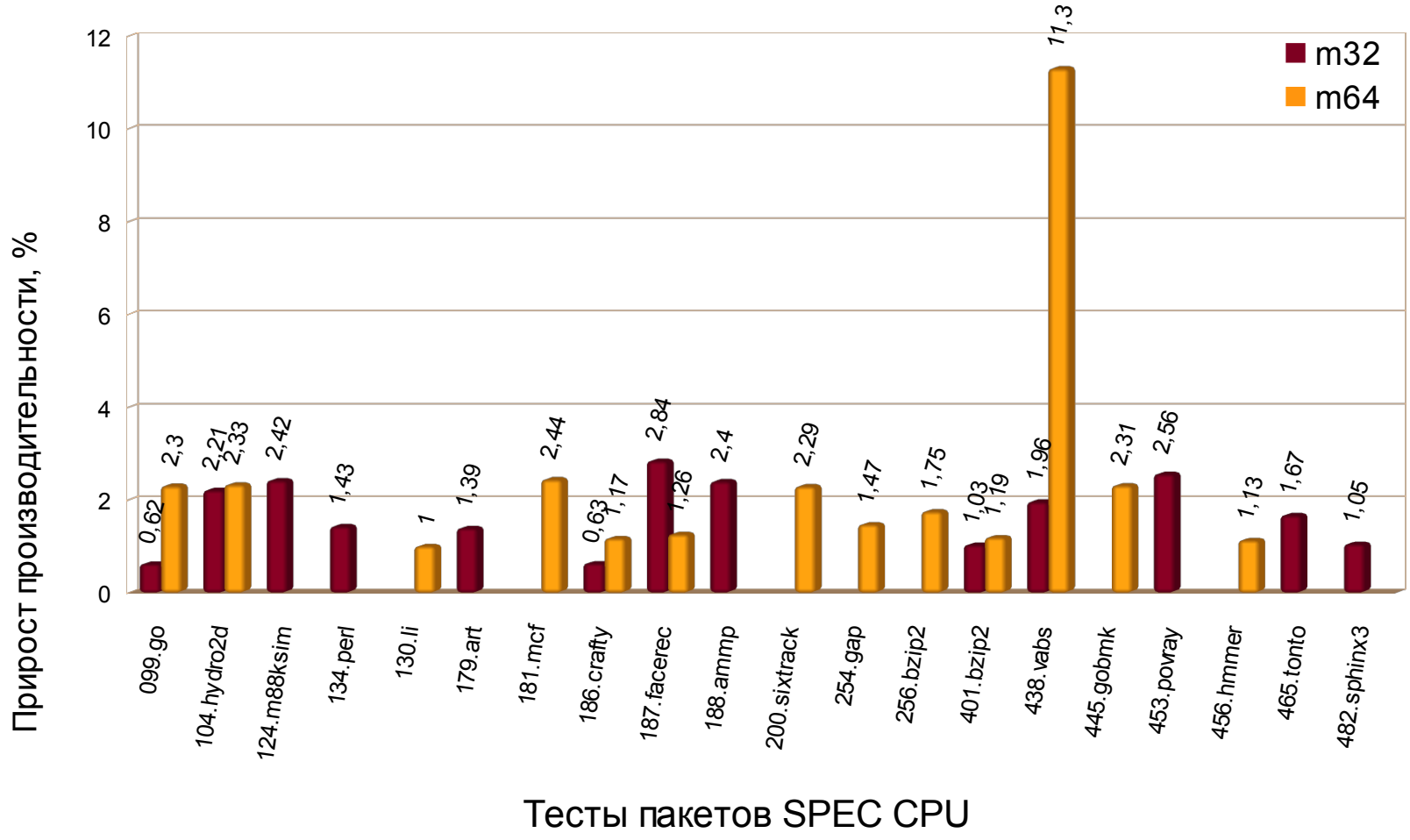
Анализ эффективности шага алгоритма конвейеризации

- Упрощенное копирование цикла
- Выполнение шага оптимизации с упрощенными корректировками
- Оценка эффективности

Адаптация для архитектуры «Эльбрус»

- Учет архитектурных особенностей:
  - количество функциональных устройств
  - типы операций
  - аппаратная поддержка оптимизаций
- Учет дополнительных аналитических структур данных:
  - Predicate Partition Graph (PPG)
- Учет некоторых специфичных оптимизаций:
  - Overlap
- Усложнение корректировок при переносе операций

Экспериментальные результаты



Результаты

- Реализована и интегрирована в оптимизирующий компилятор для архитектуры SPARC оптимизация программной конвейеризации циклов. Оптимизация адаптирована для архитектуры «Эльбрус».

- Оптимизация прошла проверку на внутренних пакетах тестирования, состоящих из отдельных задач пакетов тестирования SPEC и нескольких тысяч задач генератора случайных тестов.

- Проведены замеры эффективности оптимизации на пакетах тестирования SPEC CPU95, SPEC CPU2000, SPEC CPU2006 показавшие прирост производительности до 11%.