Разработка и применение комплекса оптимизаций для эффективного исполнения фрагментированных программ на мультикомпьютере

Ткачёва Анастасия

2018

Введение (1)

Для эффективной реализации больших задач численного моделирования на суперкомпьютере от пользователя требуется не только знание предметной области, но и знания и опыт работы в области системного параллельного программирования.

Особенно, эта проблема актуальна для задач с высокой динамикой поведения (например, приложения метода PIC в физике плазмы и астрофизике), реализация которых требует поддержки динамической балансировки нагрузки.

Введение (2)

Система фрагментированного программирования LuNA, предназначена для автоматизации процесса реализации больших задач численного моделирования на суперкомпьютере.

Проблема (1)

В системе LuNA прикладной алгоритм представляется в явной параллельной форме. Она не содержит ограничений по исполнению и распределению ресурсов, которые появляются, если мы оптимизируем исполнение под конкретный вычислитель и учитываем специфику входных данных.

Проблема (2)

В представлении ФП определено множество вариантов исполнения, автоматический выбор лучшего (в смысле времени исполнения, потребления памяти) варианта исполнения трудно-решаемая задача.

Под **поведением** будем понимать множество всех вариантов исполнения программы.

Цель работы

Разработать средства задания прямого управления для оптимизации исполнения фрагментированных программ в системе LuNA, а также алгоритмы их автоматического применения

Фрагментированная программа

Фрагментированная программа (ФП) представляет собой двудольный ориентированный граф, вершинами которого являются множество фрагментов вычислений (ФВ) и фрагментов данных (ФД), а дуги – информационные зависимости между фрагментами.

ФВ реализуется вызовом функции без побочных эффектов для некоторого набора входных ФД.

ФП исполнена, когда все ФВ исполнены. Порядок исполнения ФВ основан только на информационных зависимостях.

В системе LuNA реализуется в динамике потоковое управление (dataflow).

Runtime-система LuNA

Runtime-система LuNA является полу-интерпретатором ФП. В частности при исполнении ФП она осуществляет следующие функции:

- выделение ресурсов вычислителя для ФВ и ФД;
- выбор из множества всех ФВ множество ФВ готовых к исполнению;
- планирование вычислений (какой ФВ из множества готовых к исполнению ФВ исполнить: один, все или не одного и т.д.);
- обеспечение динамических свойств ФП (например, динамическая балансировка нагрузки).

Виды накладных расходов

1. Накладные расходы на организацию вычислений внутри узла
2. Накладные расходы на организацию распределенных вычислений
3. Накладные расходы на коммуникации

Такие накладные расходы характерны для параллельных программ, написанных для задач численного моделирования.

Анализ информационных зависимостей

Более 85% пар индексных выражение в научных программах на языке Fortran попадают в одну из четырех категорий[1]:

- 1) ZIV (zero index variables). Индексные выражения в описании ФД инвариантны для цикла выражений. Например, A[4] и A[N+3], где N не является счетчиком цикла и не зависит от него.

- 2) Strong SIV (single index variable). Пара индексных выражений имеет вид $(a \cdot i + b_1, a \cdot i + b_2)$, где i - счетчик цикла.

- 3) Weak-zero SIV. Пара индексных выражений имеет форму $(b_1, a_2 \cdot i + b_2)$ где i - счетчик цикла.

- 4) Weak-crossing SIV (шаги равны по модулю, отличаются знаком). Пара индексных выражений имеет форму $(a \cdot i + b_1, a \cdot i + b_2)$ где i - счетчик цикла.

Goff, Gina. Practical dependence testing / Gina Goff, Ken Kennedy, Chau-Wen Tseng // Proceedings of the ACM SIGPLAN conference on Programming language design and implementation. — Vol. 26. — 1991. — Pp. 15 – 29.

Оптимизация накладных расходов на организацию вычислений внутри узла

Предлагаемое решение: монолитизация множества ФВ

Идея: можно на стадии компиляции с помощью алгоритмов статического анализа выбрать и зафиксировать вариант исполнения множества ФВ, не противоречащий информационным зависимостям.

Из этого следует, что ***монолитным ФВ*** будем называть такой ФВ, который с точки зрения runtime-системы атомарный, но внутри себя заключает исполнения нескольких ФВ, причем порядок исполнения статически фиксирован, не противоречит информационных зависимостях между ФВ, но при этом лишен недетерминизма.

Применение:

- Мелкая фрагментация данных или слишком много параллелизма
- Вычисления над распределенными массивами
- Редуцирование данных

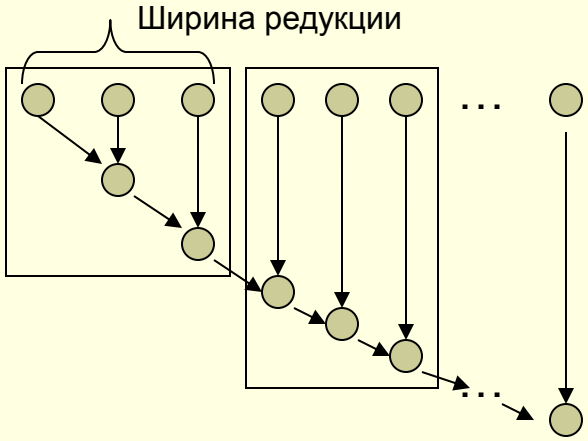
Реализация: монолитизация ФВ

Разработаны и встроены в компилятор LuNA модули, обеспечивающие монолитизацию множества ФВ, задаваемых через циклические конструкции и в виде подпрограммы ФП.

Задачи для тестирования

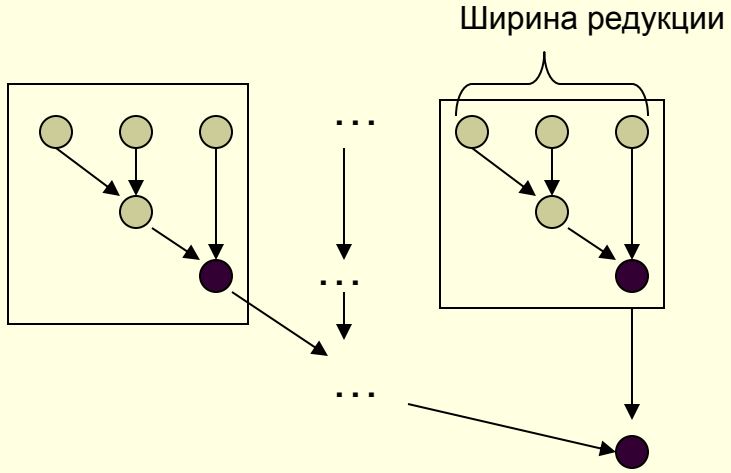
Тест 1

Последовательный алгоритм редуцирования данных



Тест 2

Параллельный алгоритм редуцирования данных



Параметры вычислителей

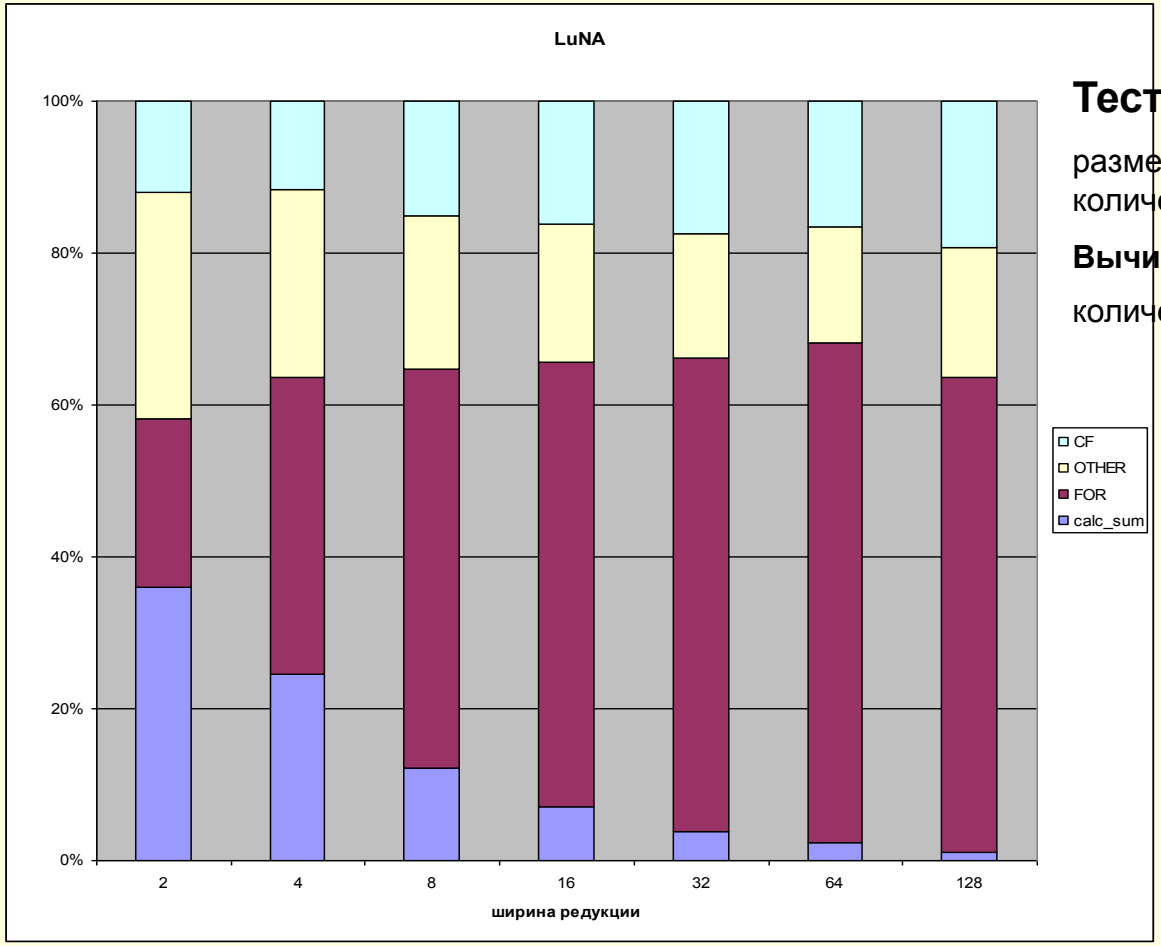
Вычислитель 1

Компьютер с 6 ядрами (Intel Xeon CPU X5600 2,8 Ghz)

Вычислитель 2

Кластер НГУ

Процентное соотношение (%) исполнения ФП базовым алгоритмом runtime-системы LuNA



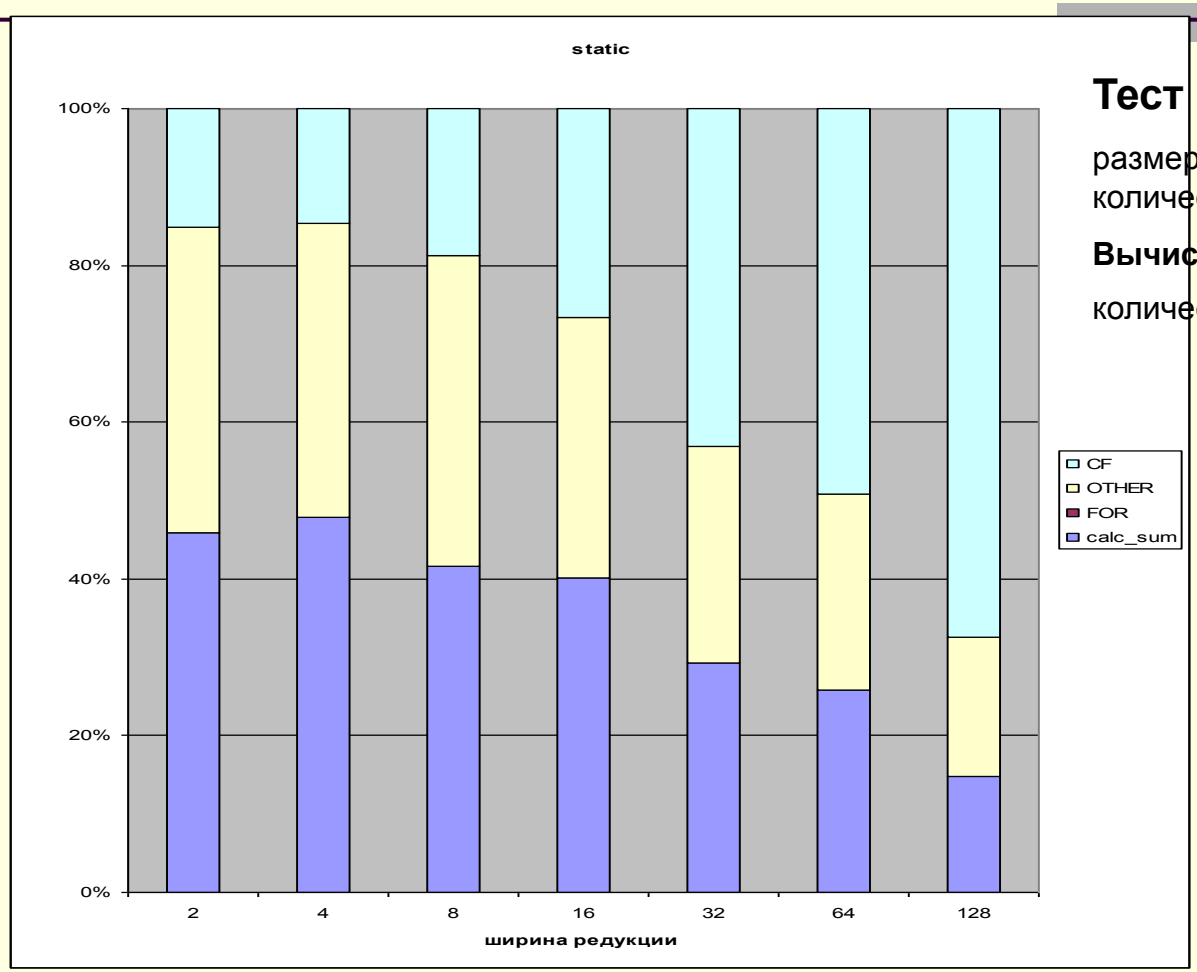
Тест 1

размер ФД 10x10x10,
количество редукций 200,

Вычислитель 1

количество потоков 4

Процентное соотношение (%) исполнения ФП с использованием директивы static



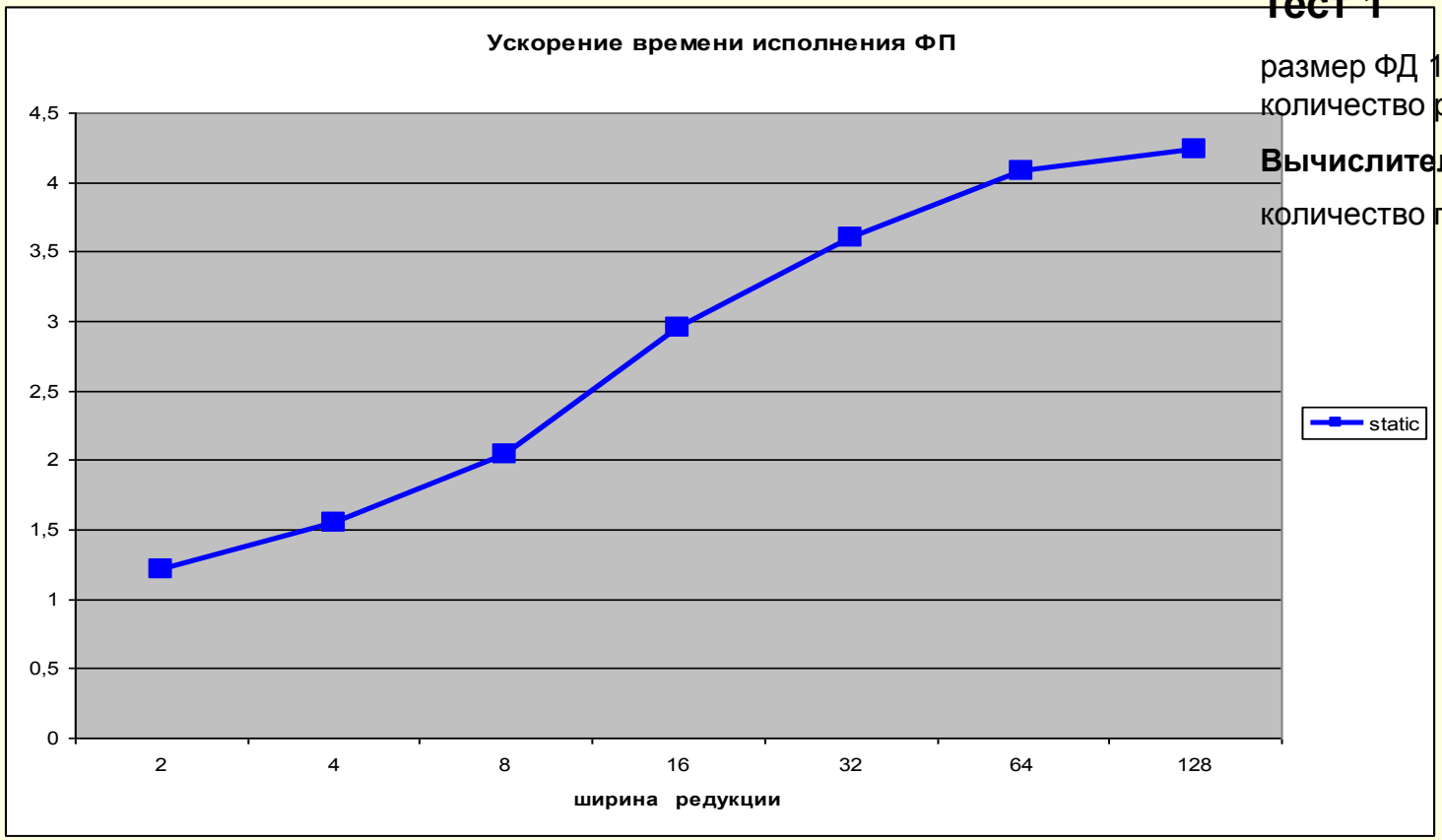
Тест 1

размер ФД 10x10x10,
количество редукций 200,

Вычислитель 1

количество потоков 4

Выигрыш от использования директивы static в зависимости от ширины редукции



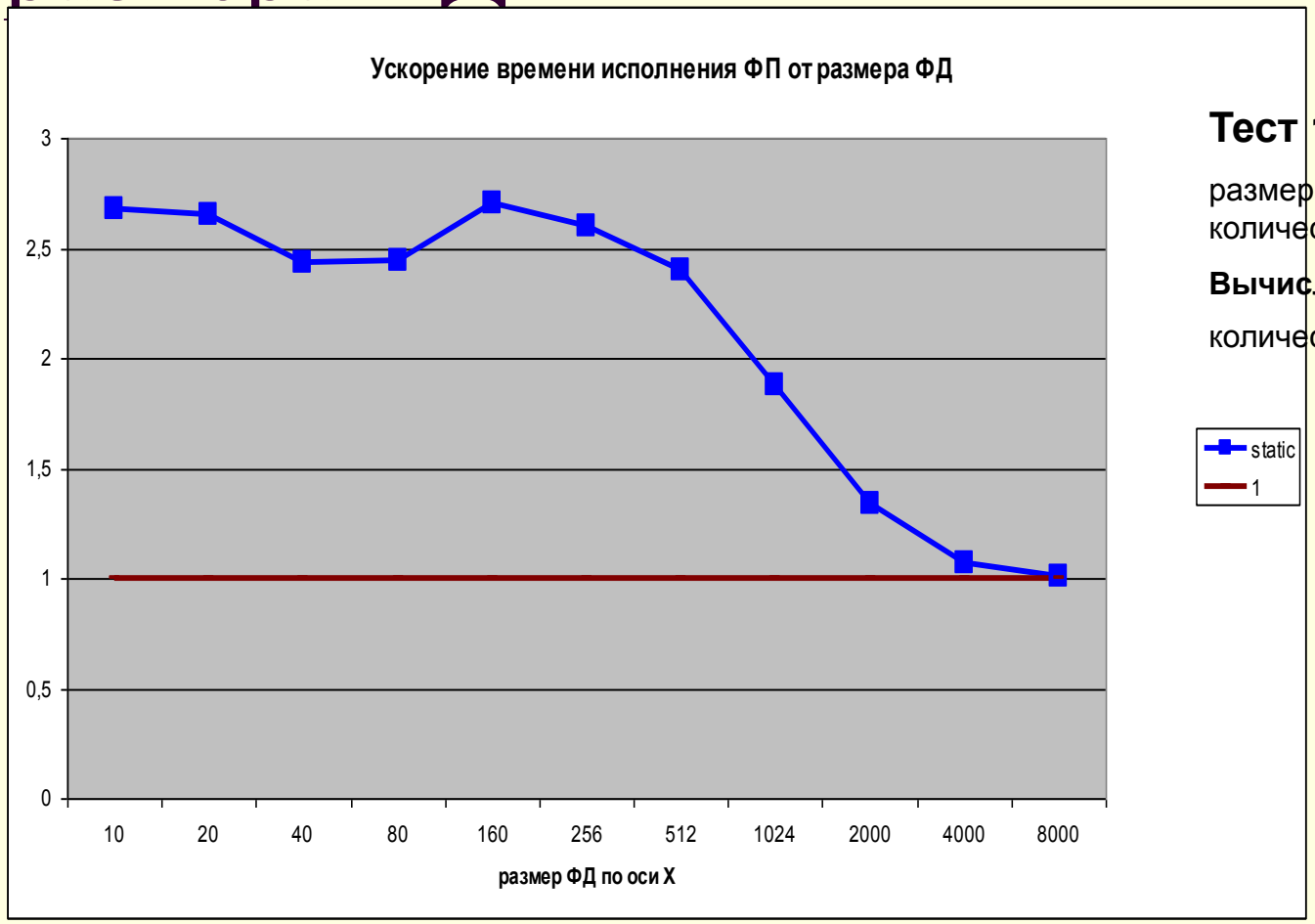
Тест 1

размер ФД 10x10x10,
количество редукций 200,

Вычислитель 1

количество потоков 4

Выигрыш от использования директивы static в зависимости от размера ФД



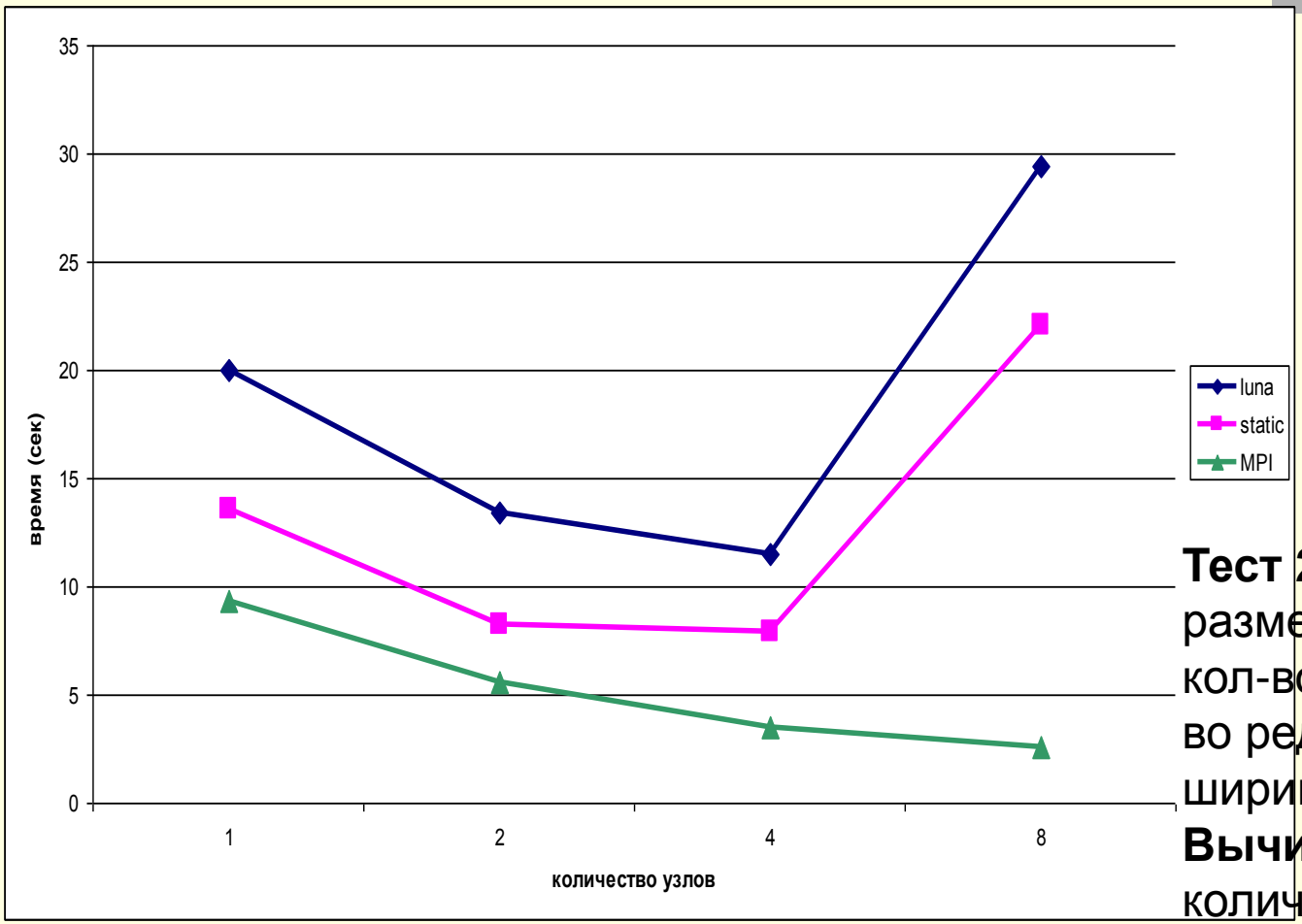
Тест 1

размер ФД 10x10x10,
количество редукций 200,

Вычислитель 1

количество потоков 4

Тестирование в распределенной памяти



Тест 2
размер ФД 40x40x40, кол-во ФВ 10000, кол-во редукций 10, ширина редукции 250
Вычислитель 2
количество потоков 4

Предлагаемые средства: предикатная сеть Петри

В качестве средств для задания прямого управления была выбрана предикатная сети Петри.

Почему сеть Петри:

- широко-известная модель, её свойства хорошо изучены;
- с помощью неё можно задавать управление для широкого класса задач;

Интерпретация предикатной сети Петри:

- Срабатывание перехода — срабатывание некоторого ФВ
- Нахождение фишки в месте означает, что значение некоторого ФД вычислено.
- Условие срабатывание переходя – условие срабатывания ФВ

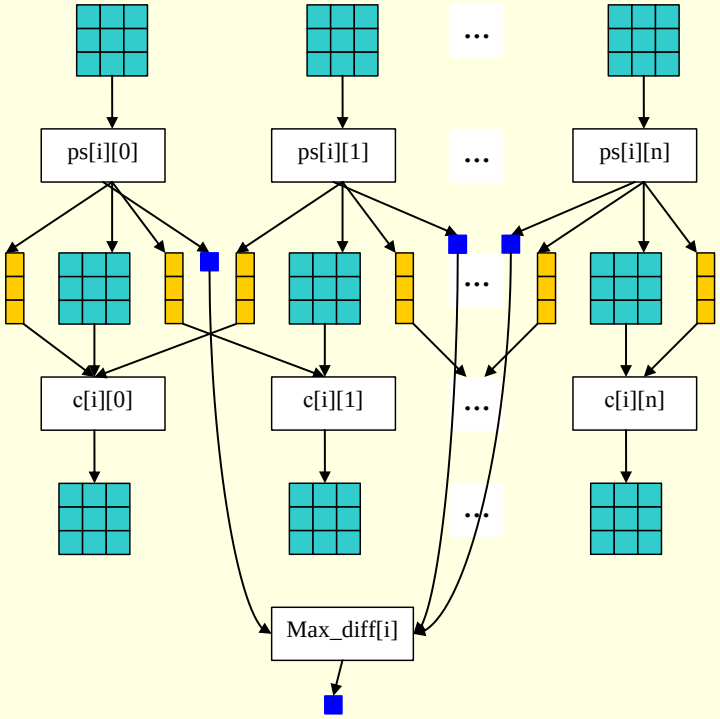
Порядок выполнения ФВ определяется функционированием сети Петри.

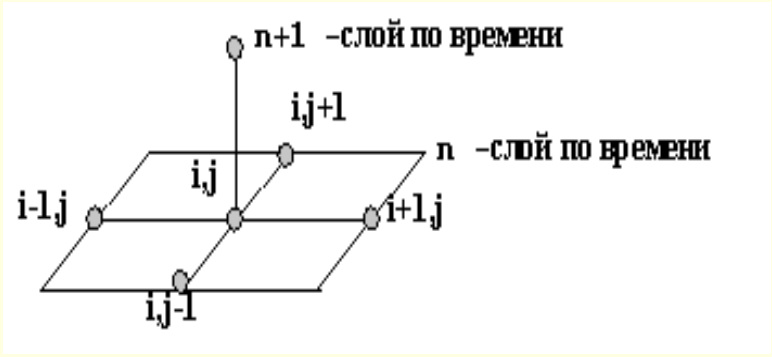
Реализация

Был реализован программный модуль RuSh, который обеспечивает параллельное выполнение ФП в общей памяти, в которой порядок исполнения ФВ задан предикатной сетью Петри.

Разработаны и встроены в компилятор LuNA модули, генерирующие предикатную сеть Петри на основе статического анализа, для задаваемого подмножества ФВ.

Решение уравнения Пуассона явным методом (i-итерация 1D декомпозиция)





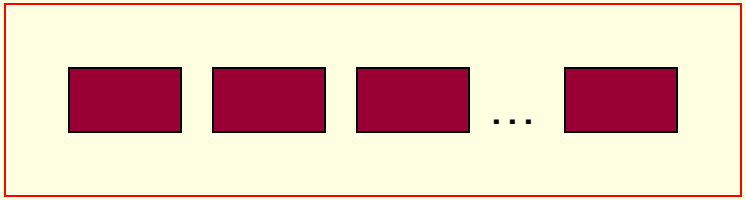
Исследование производительности

- Параметры вычислителя
  - Intel Xeon CPU X5600 2.8Ghz (6 ядер)
- Параметры задачи
  - 3D декомпозиция

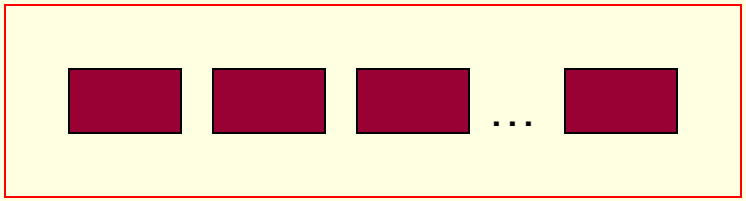
    (размер пространства 100x100x100, iter 40)

Варианты тестов

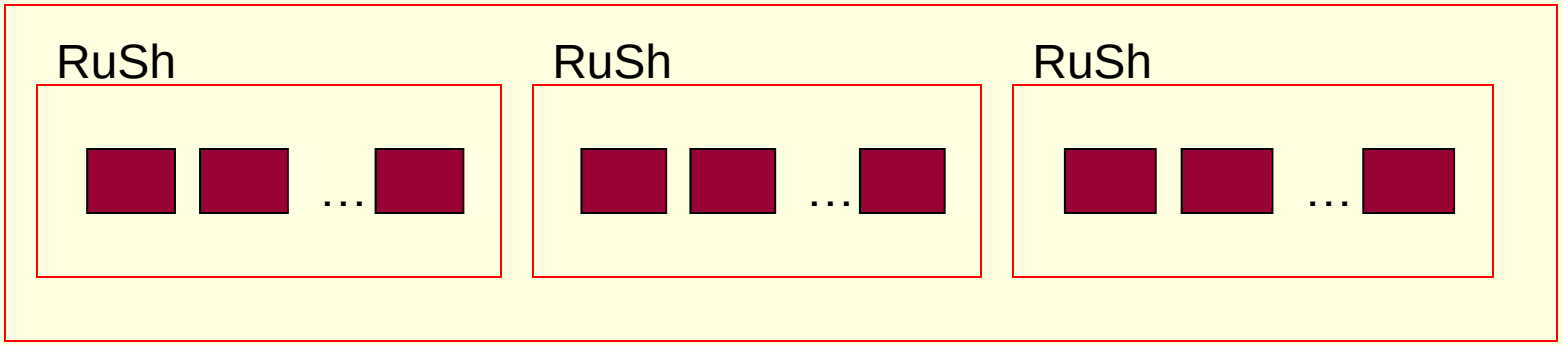
LuNA



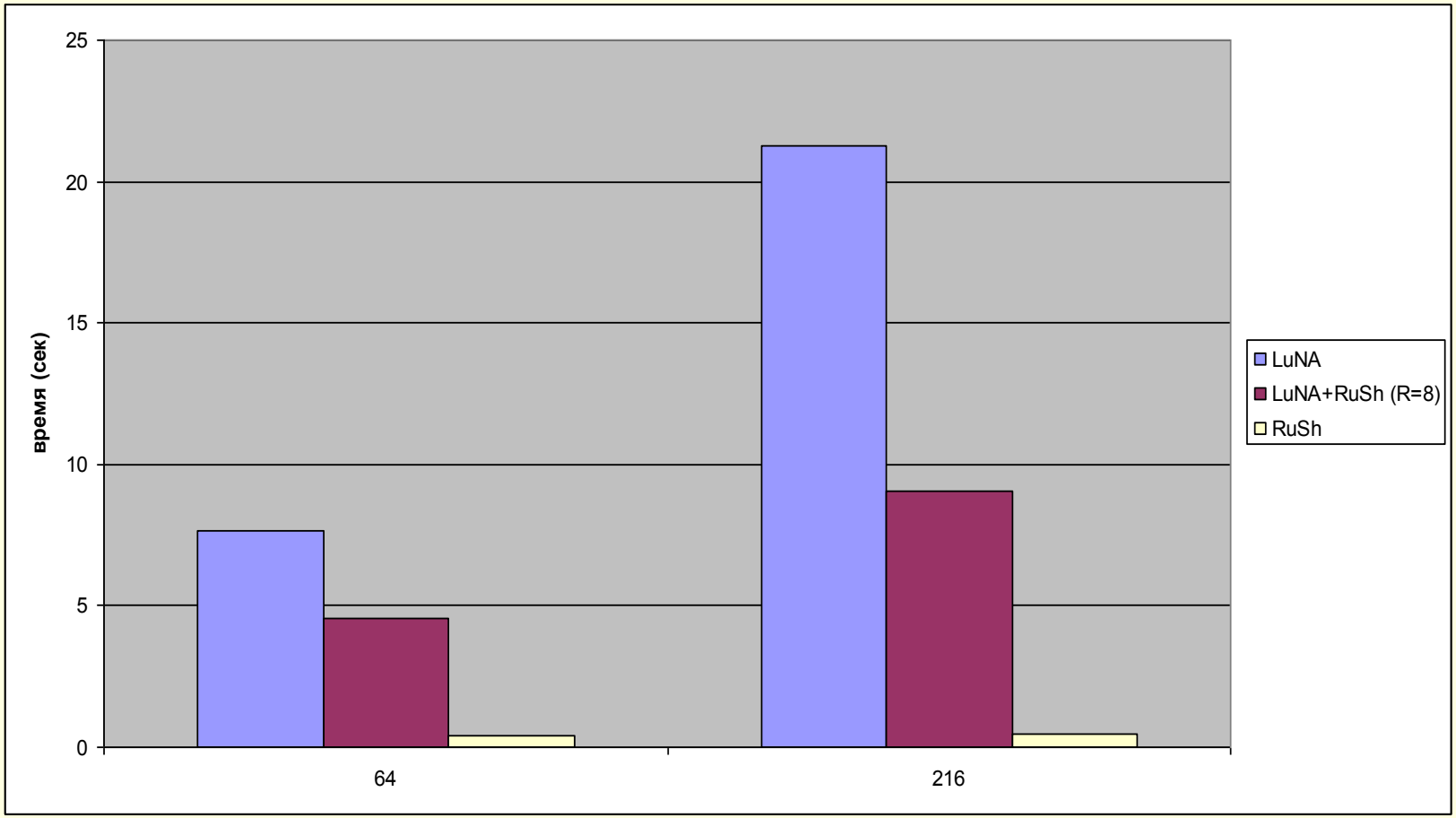
RuSh



LuNA



Сравнение времени работы LuNA и RuSh: 3D декомпозиция

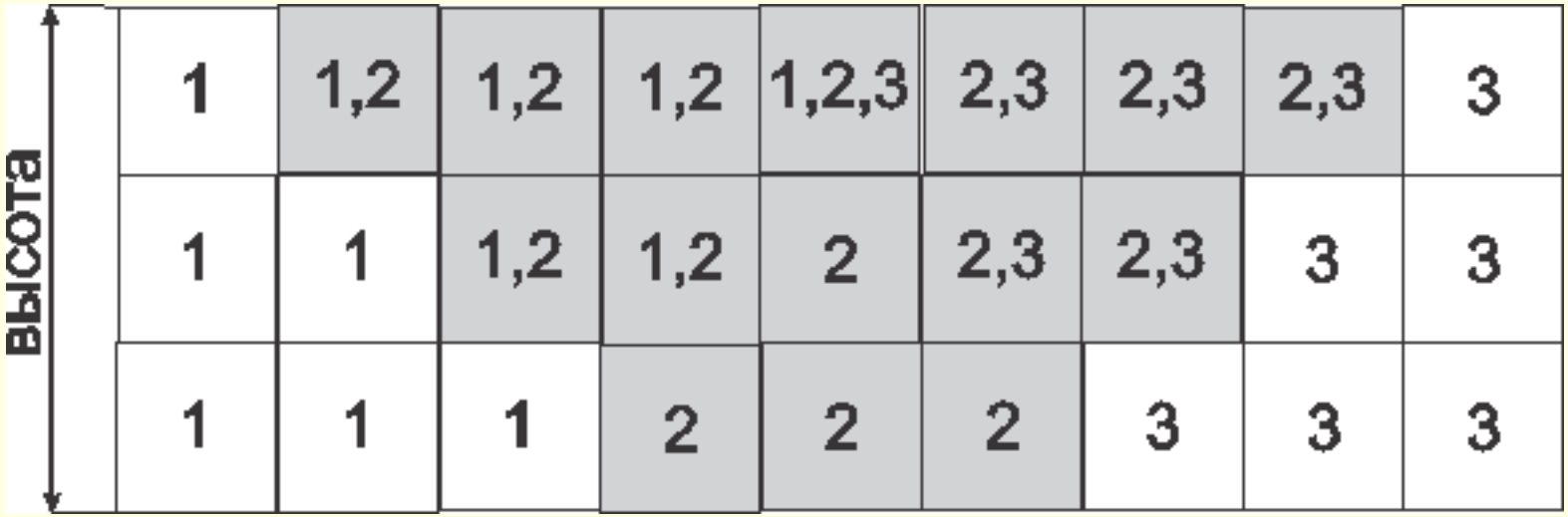


Тестирование в распределенной памяти

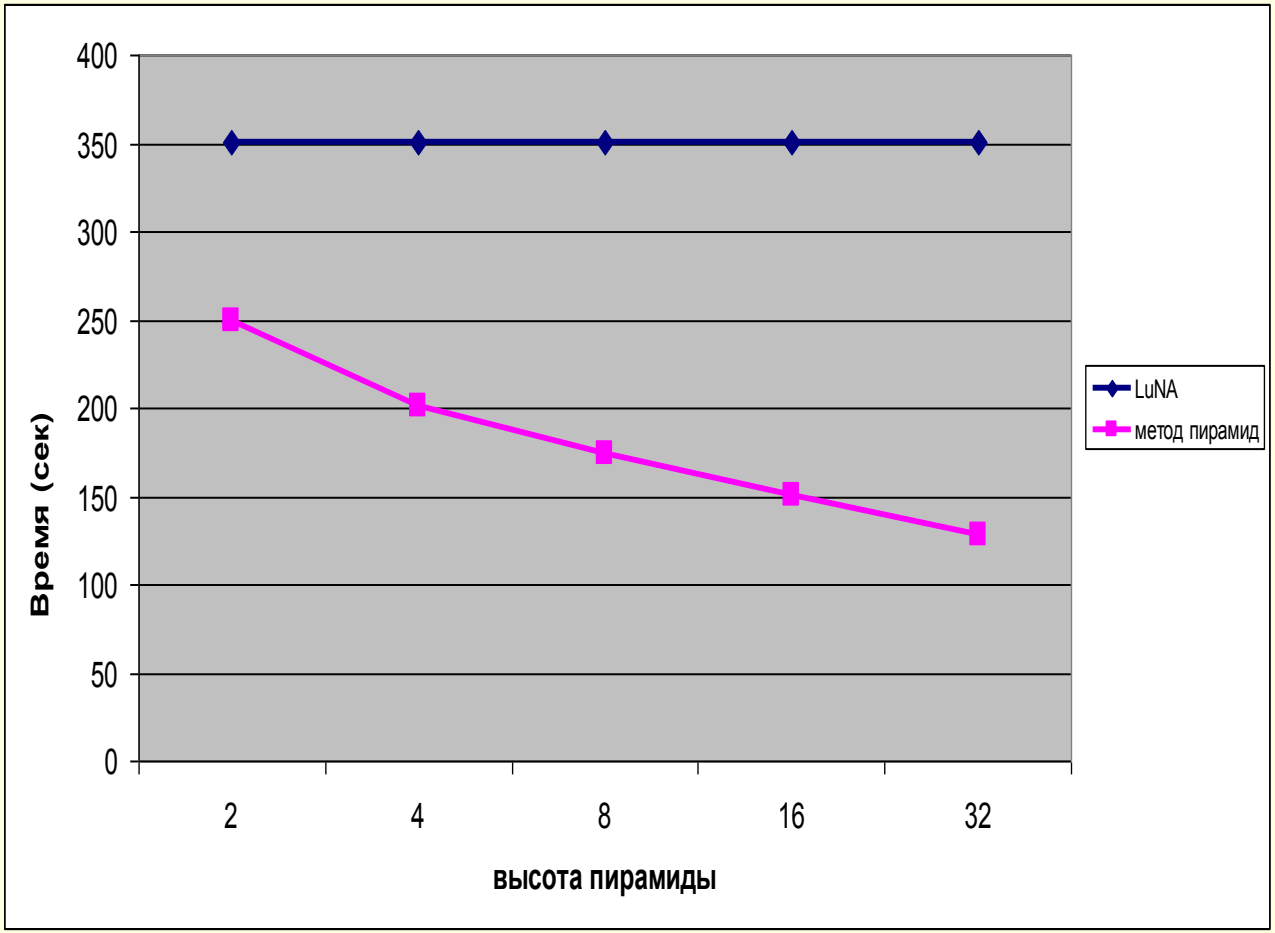
Применение метода пирамид к задаче решения уравнения Пуассона явным методом

Пространство итераций разбивается по переменной, связанной с итерационным шагом прикладной задачи на равные интервалы, к которым применяется метод пирамид

Пример разбиения методом пирамид внутри одного интервала



Тестирование



Параметры задачи:

2000x200x200, кол-во ФВ 256, кол-во итераций 128.

Кластер КазНУ им. аль-Фараби,

г.Алма-Ата, Казахстан,

MPI-процессов 4

Оптимизация накладных расходов на организацию распределенных вычислений

Управляющая программа

Управляющая программа, которая представляет собой C++ класс для исполнения в распределенной и общей памяти.

Следующие обработчики событий в управляющей программе должны быть определены:

- **onInit ()** – функция начального значения для начальной инициализации.
- **onComputed (df_id)** – функция, вызываемая после того как каждый ФД с идентификатором df_id вычислен.
- **onReceived (df_id)** – функция, вызываемая после того как каждый ФД с идентификатором df_id получен от другого узла.
- **onCfFinished (cf_id)** – функция, вызываемая после того как каждый ФВ с идентификатором cf_id завершил исполнения.

Действия, поддерживаемые LuNAFW

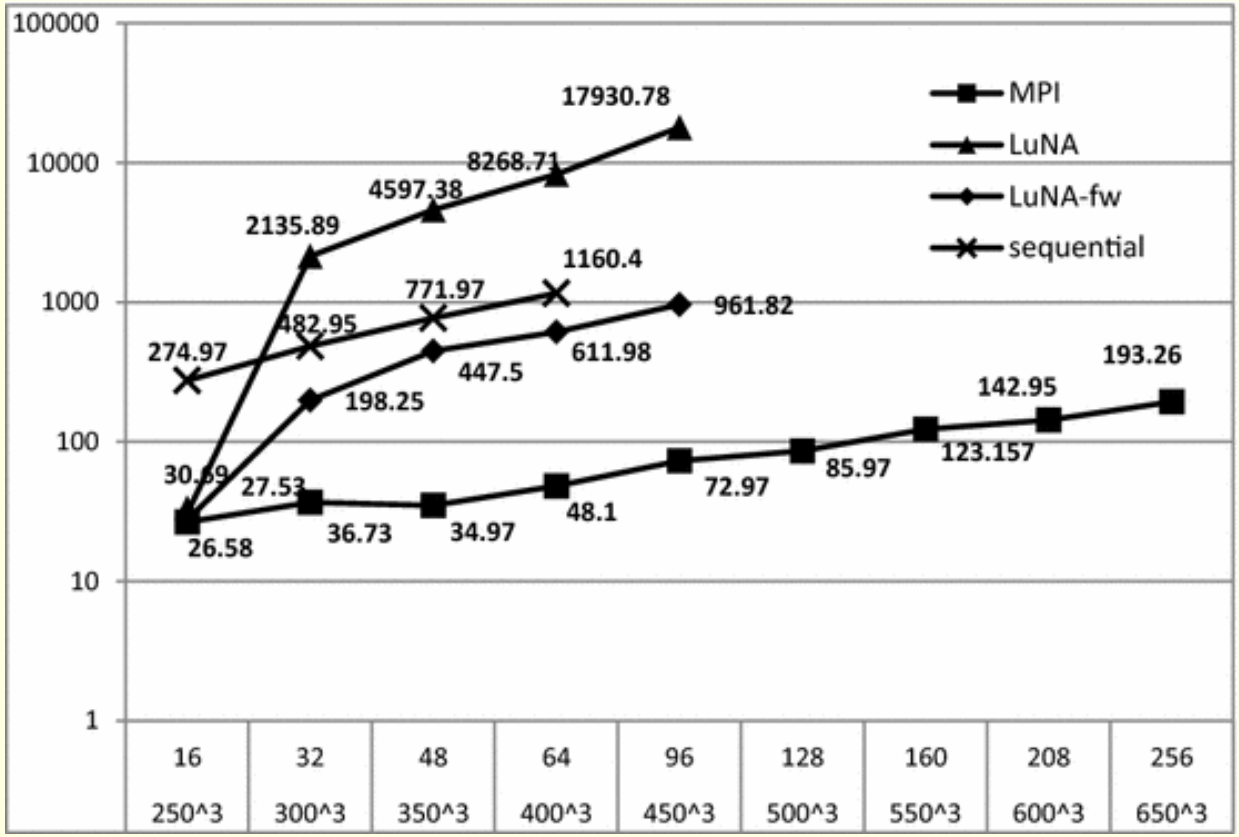
Внутри выше описанных функций могут быть вызваны следующие функции (действия), поддерживаемые фреймворком LuNAFW:

- **startCF** (CF description) – запустить исполнение ФВ.
- **checkCF** (CF description) – если все ФД доступны, то вызвать функцию
- **startCF** для CF.
- **destroyDF**(df_id) – удалить ФД с идентификатором df_id.
- **sendDF** (df_id, rank) - послать ФД с идентификатором df_id на процесс rank.
- **exit()** – завершить работу фреймворка.
- **getRank**(identifier) - функция, выдающая номер процесса, на который ФВ распределен.

Тестирование с управляющими программами, написанными вручную

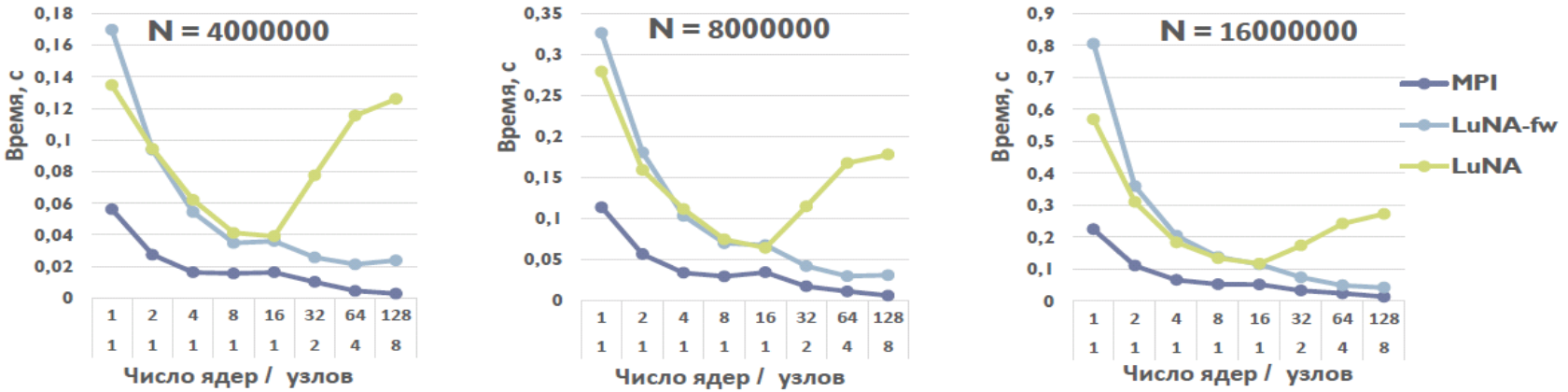
Задача решение краевой задачи фильтрации трехфазной жидкости (нефть-вода-газ), mvs10p

- [1]Akhmed-Zaki D.Z., Lebedev D.V., Perepelkin V.A. (2017) Implementation of a three dimensional three-phase fluid flow (“oil–water–gas”) numerical model in LuNA fragmented programming system. Journal of Supercomputing, Volume 73, Issue 2, pp 624–630.



Тестирование:
Сравнение различных реализаций

IADE метод



- Выводы
  - В пределах узла варианты LuNA и LuNA-fw имеют сравнимую производительность
  - При использовании более одного узла производительность варианта LuNA падает
  - Вариант MPI обгоняет LuNA-fw примерно в 2 раза

Киреев С.Е., Перепелкин В.А., Ткачёва А.А. Реализация метода IADE_RB_CG в системе фрагментированного программирования LuNA // Вос. Сиб. конф. по параллельным и высокопроизводительным вычислениям: труды конф.,Томск: изд-во Том.Ун-та,2015,с.66-72.

Реализация

Был разработан алгоритм генерации управляющих программ для ФП или подпрограмм ФП на основе статического анализа ФП, порядок исполнения ФВ не противоречит информационным зависимостям, а распределение ресурсов задается на основе рекомендаций пользователя

Тестирование со сгенерированными управляющими программами

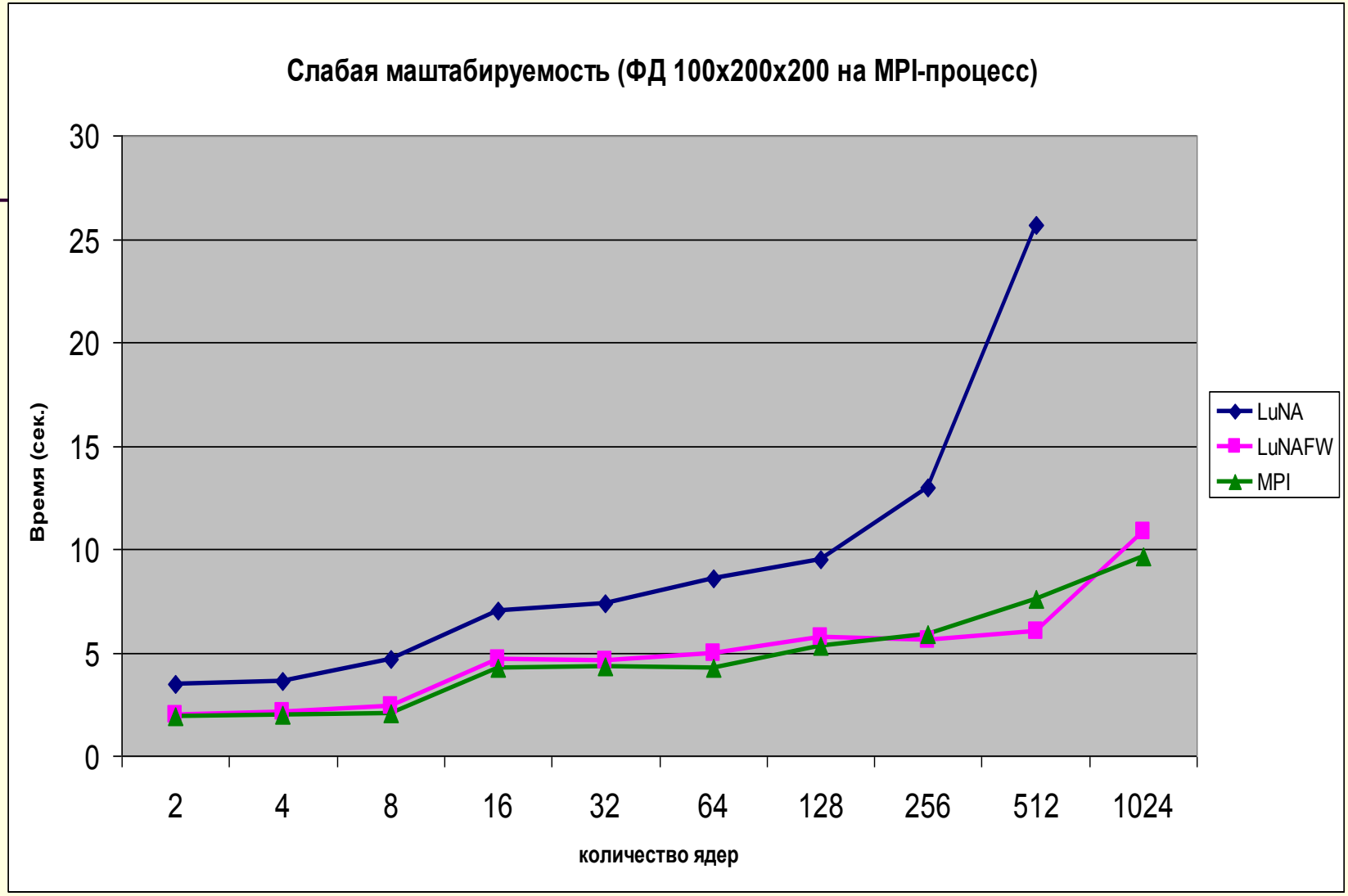
Условия запуска

Вычислитель: mvs10p (16 ядер на узел)

- 1 поток в системе LuNA.
- 1 MPI процесс на ядро.
- 1 ФВ на 1 MPI процесс.

Параметры задачи:

- Количество итераций 20.
- Размер ФД 100x200x200 на 1 MPI процесс


Слабая маштабируемость (ФД 100x200x200 на MPI-процесс)
Время (сек.)
30
25
20
15
10
5
0
2
4
8
16
32
64
128
256
512
1024
количество ядер
LuNA
LuNAFW
MPI

Результаты

Автоматически сгенерированная управляющая программа позволила улучшить производительность исполнения ФП на 40% по сравнению с базовым алгоритмом исполнения ФП run-time системы LuNA.

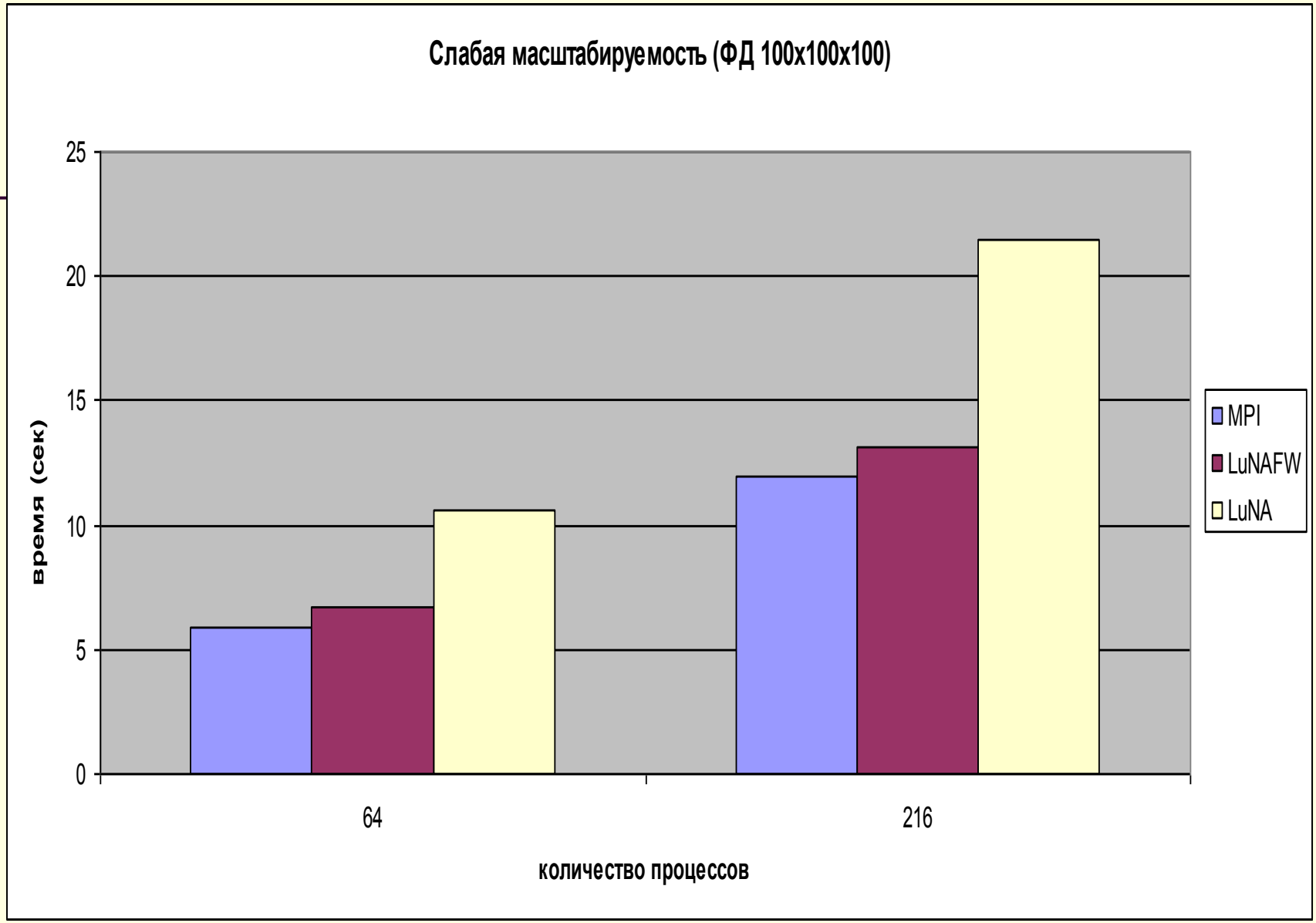
Решение уравнения Пуассона явным методом 3D декомпозиция

Вычислитель: mvs10p (16 ядер на узел)

- 1 поток в системе LuNA.
- 1 MPI процесс на ядро.
- 1 ФВ на 1 MPI процесс.

Параметры задачи:

- Размер ФД 100x100x100.
- Количество итераций 10.


Слабая масштабируемость (ФД 100x100x100)
время (сек)
25
20
15
10
5
0
64
216
количество процессов
MPI
LuNAFW
LuNA

Практическая ценность

Разработанные средства задания прямого управления могут использоваться для оптимизаций исполнения программ, представляемых в декларативной форме, а именно в языках и системах, где в программе параллелизм задается в явной форме. Результаты работы успешно используются в компиляторе системы LuNA.

Заключение

Разработаны, реализованы и встроены в систему LuNA средства задания прямого управления для оптимизации накладных расходов на организацию вычислений внутри узла в виде монолитизации множества ФВ и предикатной сети Петри. Оба средства применяются в ФП над распределенными массивами данных, могут задаваться с помощью директив пользователя и автоматически с помощью статического анализа.

Разработан алгоритм генерации управляющих программ, в которых на основе событийно-ориентированной модели вычислений, задаются частичные решения о распределении ресурсов и порядке исполнения операций. Показано, что использование сгенерированных управляющих программ, позволяет получить приемлемую производительность исполнения ФП, сравнимую с MPI на явном методе решения уравнения Пуассона при одномерной и трехмерной декомпозиции данных. Сам алгоритм генерации рассчитан и может быть применен для фрагментированных программах, информационные зависимости определяются SIV-тестом.

Спасибо

за внимание!

Анализ профиля исполнения ФП

Для определения степени фрагментации предлагается использовать следующую метрику:

$$g(FP) = \min_{i=1:v} \min\left(\frac{t_i}{\max_{j \in pred(i)} c_{j,i}}, \frac{t_i}{\max_{j \in succ(i)} c_{i,j}}\right)$$

Анализируется граф, вершины которого ФВ, ребра информационные зависимости между ФВ, t_i - время вычисления ФВ, c_{ij}- время коммуникаций между i-ФВ и j-ФВ. Будем говорить, что ФП с мелкой фрагментацией данных, если . Метрику можно применять не ко всей ФП, а некоторому подмножеству ФВ. Она говорит о том, что время вычислений меньше времени, требующегося на коммуникации.