ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Разработка и реализация переносимых алгоритмов распределенного исполнения фрагментированных программ

Выполнил: **Ажбаков Артем Альбертович**

Научный руководитель: **Малышкин Виктор Эммануилович**

д.т.н., зав. кафедрой, кафедра ПВ ФИТ

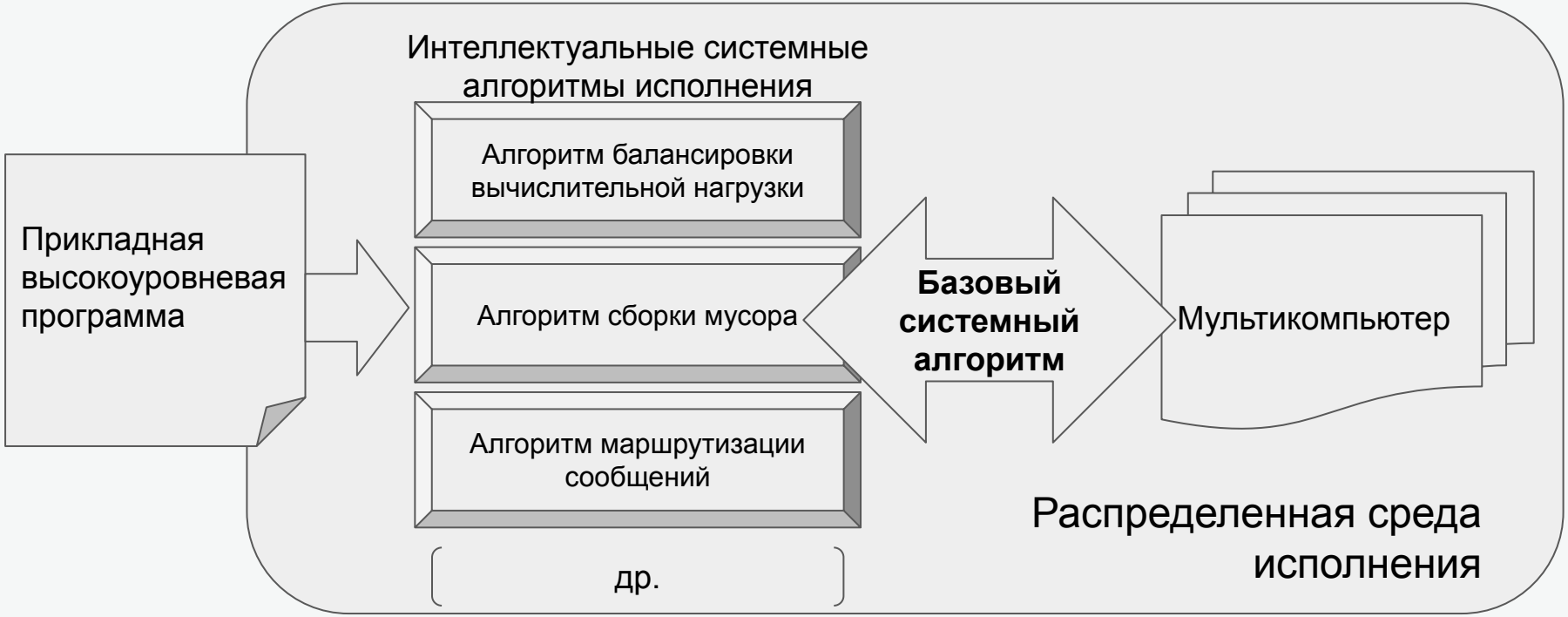
Проблема интеграции неоднородных вычислительных ресурсов в численном моделировании

Тенденция роста неоднородности вычислительных ресурсов: добавление/замена кластерного оборудования, перспективы объединения вычислительных сетей, перспективы использования неспециализированных устройств.

При выполнении параллельной программы на неоднородных вычислительных ресурсах особую сложность обретают задачи:

- Динамическая балансировка нагрузки
- Эффективная передача данных
- Сборка мусора
- Отказоустойчивость
- Другие

Системы автоматизации параллельного программирования



Необходима программная платформа для экспериментальных исследований системных алгоритмов параллельного исполнения программ.

Обзор: существующие системы автоматизации параллельного программирования

Для исследовательской платформы важно:

- Переносимость
- Модульность программной архитектуры
- Масштабируемость
- Модель вычислений

Charm++, OpenTS, SMP Superscalar, PaRSEC:

+ применяется фрагментирование, поддерживаются неоднородные вычислители, осуществляется динамическая балансировка нагрузки мультикомпьютера
- слабая поддержка использования статических свойств алгоритма

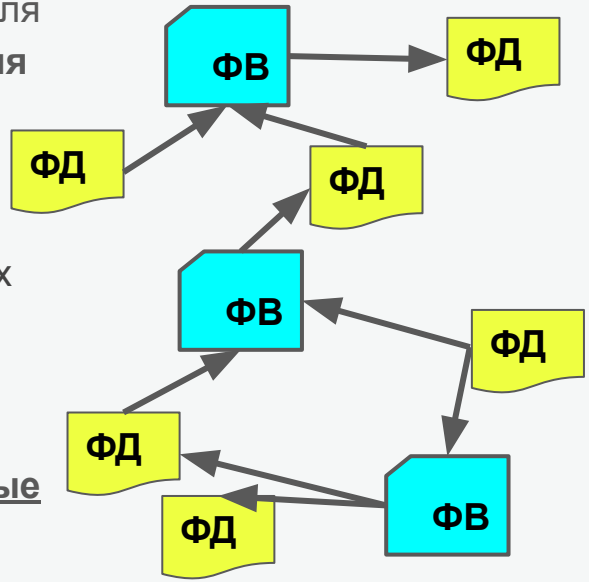
Обзор: система фрагментированного программирования LuNA

Фрагментированное представление алгоритма дает возможность среде исполнения анализировать статические свойства алгоритма, располагает к использованию разнородных вычислительных ресурсов.

Система LuNA **подходит** для использования в качестве платформы для практических исследований, однако существующая **среда исполнения на C++ слабо переносима.**

Имеется потенциал задействовать обширный класс мобильных вычислительных устройства для создания гораздо более разнородных вычислительных сетей для исследований системных алгоритмов, чем возможно на данный момент.

Чтобы задействовать имеющиеся неоднородные вычислительные ресурсы, нужно создать переносимую среду исполнения.



Цель работы

- ***разработка переносимой распределенной среды исполнения на базе web-технологий для системы фрагментированного программирования задач численного моделирования LuNA.***

Требования: переносимость, модульность, масштабируемость.

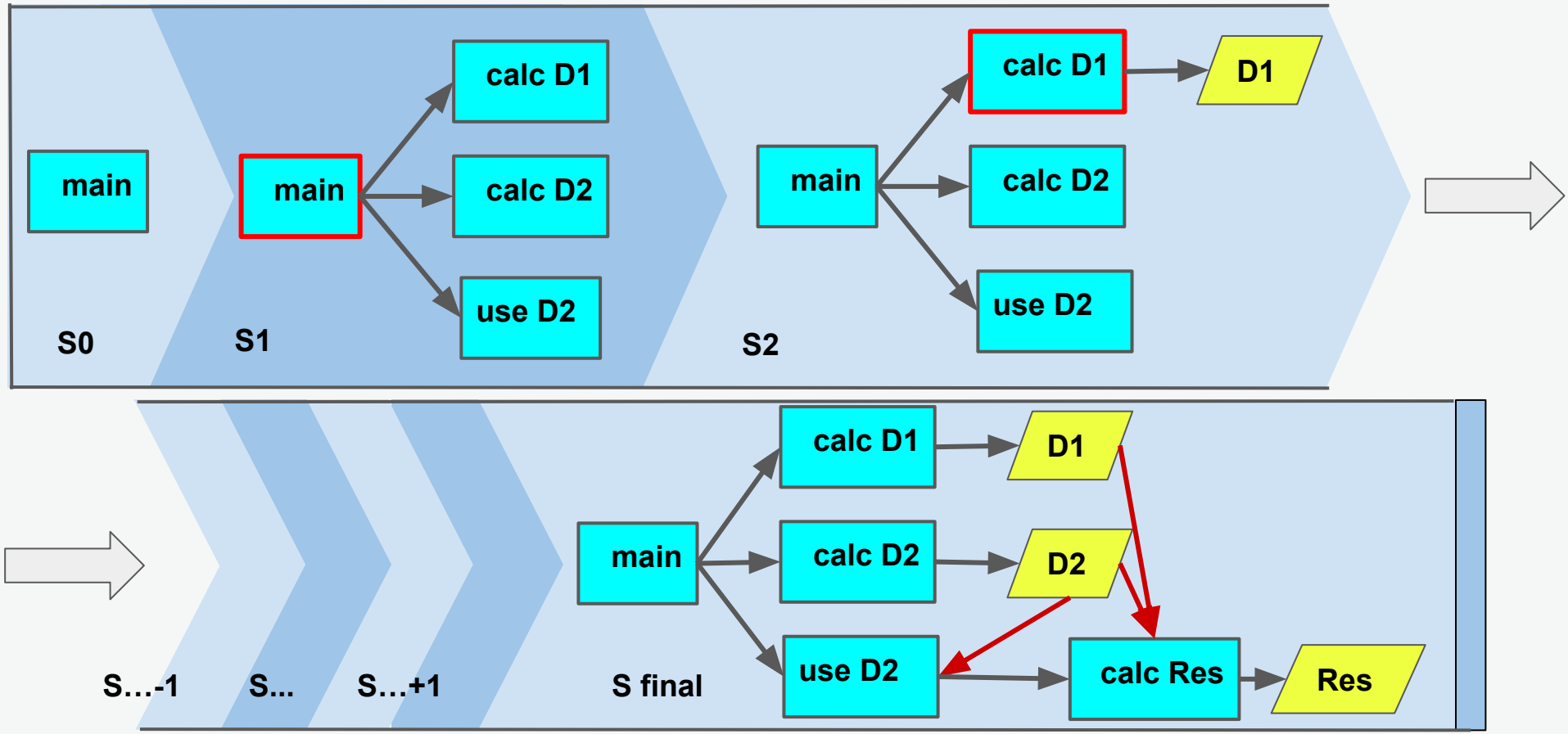
Выбор средств: web-технологии (JavaScript Runtime Environment)

Задачи

Для достижения цели были поставлены следующие задачи:

- Сформулировать модель исполнения фрагментированной программы;
- Проанализировать модель исполнения с целью выявления функций среды исполнения LuNA и их особенностей;
- Разработать модульную архитектуру среды исполнения и алгоритм ее работы;
- Реализовать программно в соответствии с требованиями.

Модель исполнения фрагментированной программы



Функции среды исполнения

1. Выбор топологии вычислительной сети;
2. Выбор коммуникационных протоколов и технологий передачи данных между узлами вычислительной сети;
3. Начальное расположение оператора вызова подпрограммы main;
4. Выбор узлов мультикомпьютера для новых операторов и фрагментов данных:
   - 4.1. Использовать ли системные коммуникаций для получения информации о состоянии окружающих узлов при выборе узла-получателя;
   - 4.2. Выбор узлов, с которыми осуществляются системные коммуникации;
   - 4.3. Организация кэширования фрагментов данных на нескольких узлах мультикомпьютера;
5. Выбор алгоритма сборки мусора (освобождение памяти использованных фрагментов данных):
   - 5.1. Алгоритм распределенной сборки мусора при использовании кэширования фрагментов данных, протокол сопутствующих системных коммуникаций;
6. При обеспечении отказоустойчивости (логическое замещение вычислительного узла при аварийном отключении):
   - 6.1. Дублирование фрагментов данных и операторов, избыточное вычисление, алгоритм голосования при определении результата избыточного исполнения оператора;
7. При выполнении операторов:
   - 7.1. Способы организации параллельного исполнения операторов в пределах одного узла мультикомпьютера;
   - 7.2. Стратегии раскрутки циклов;

Архитектура узла среды исполнения:

Программная реализация

Сервер

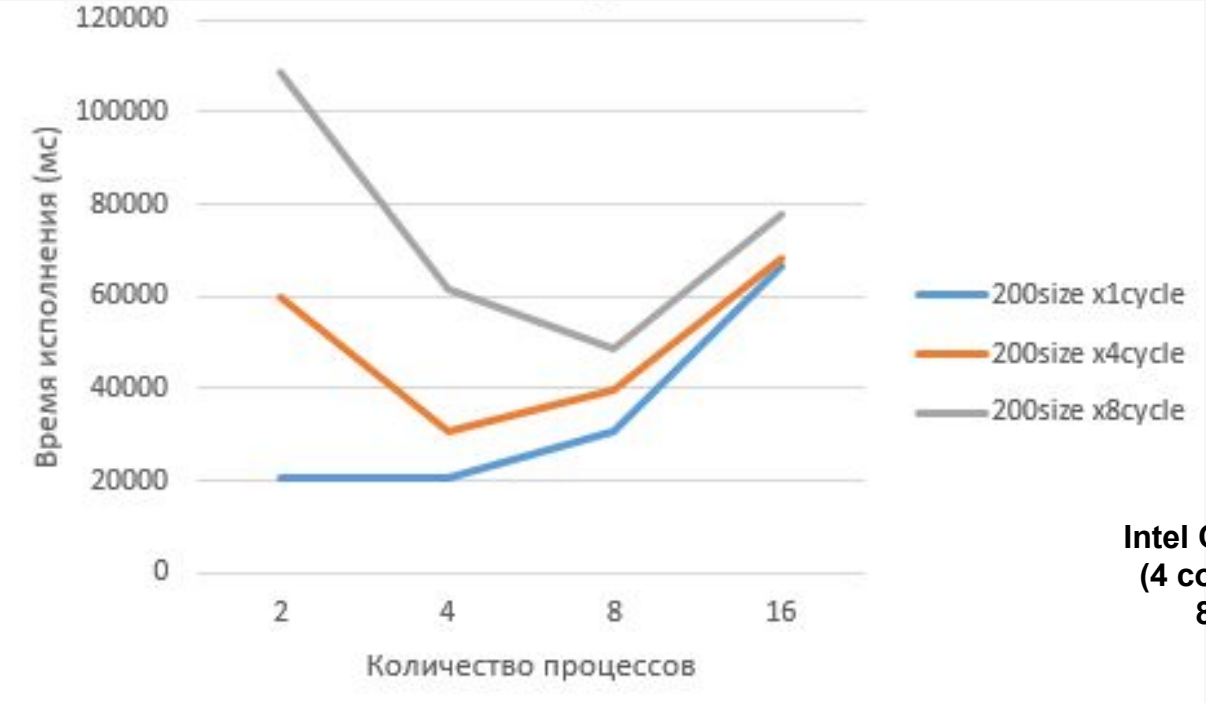
- Выдача страниц
- Создание/настройка вычислительных задач
- Проектирование представления для загружаемых топологий
- Связывание узлов (signaling)
- Загрузка и хранение пользовательских ресурсов, доступ к ним

Клиентское приложение

- Загрузка файлов на сервер
- Создание, конфигурирование, подключение к вычислительной задаче
- Связывание узлов (установление p2p-связи)
- Исполнение LuNA-программ
- Пользовательский интерфейс (разметка LuNA-программы, просмотр LuNA-фрагментов, состояние соединения с соседями, управление исполнением)

Тестирование

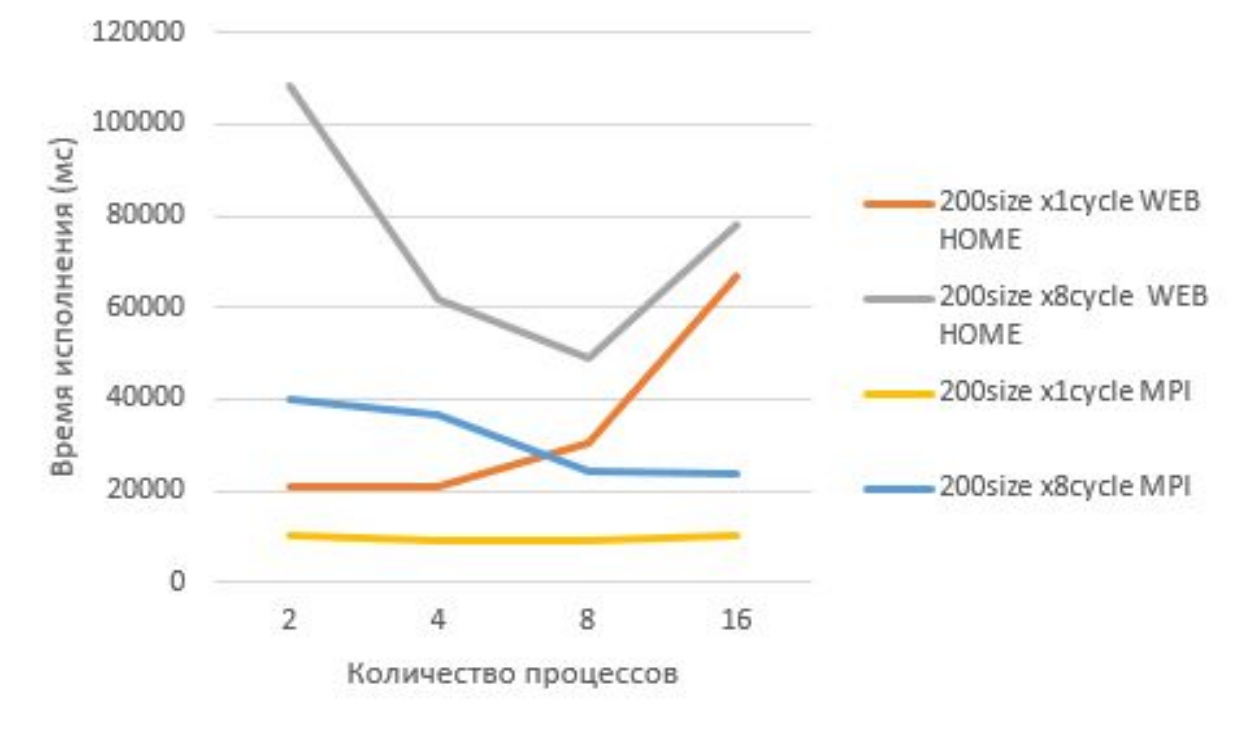
Тест: решение уравнения Пуассона методом Якоби в трехмерной сетке размером 200x200x200, 85 итераций
Топология сети: одномерная решетка



Intel Core i5-7600K (4 core, 3.8 Ghz), 8Gb RAM

Тестирование

Тест: решение уравнения Пуассона методом Якоби в трехмерной сетке размером 200x200x200, 85 итераций
Топология сети: одномерная решетка



Web:
Intel Core i5-7600K (4 core, 3.8 Ghz), 8Gb RAM

MPI:
Intel Core i7-4790K (4 core, 4.0 Ghz), 32Gb RAM

Заключение, защищаемые положения:

Создана программная платформа для исследования системных алгоритмов распределенного исполнения параллельных программ.

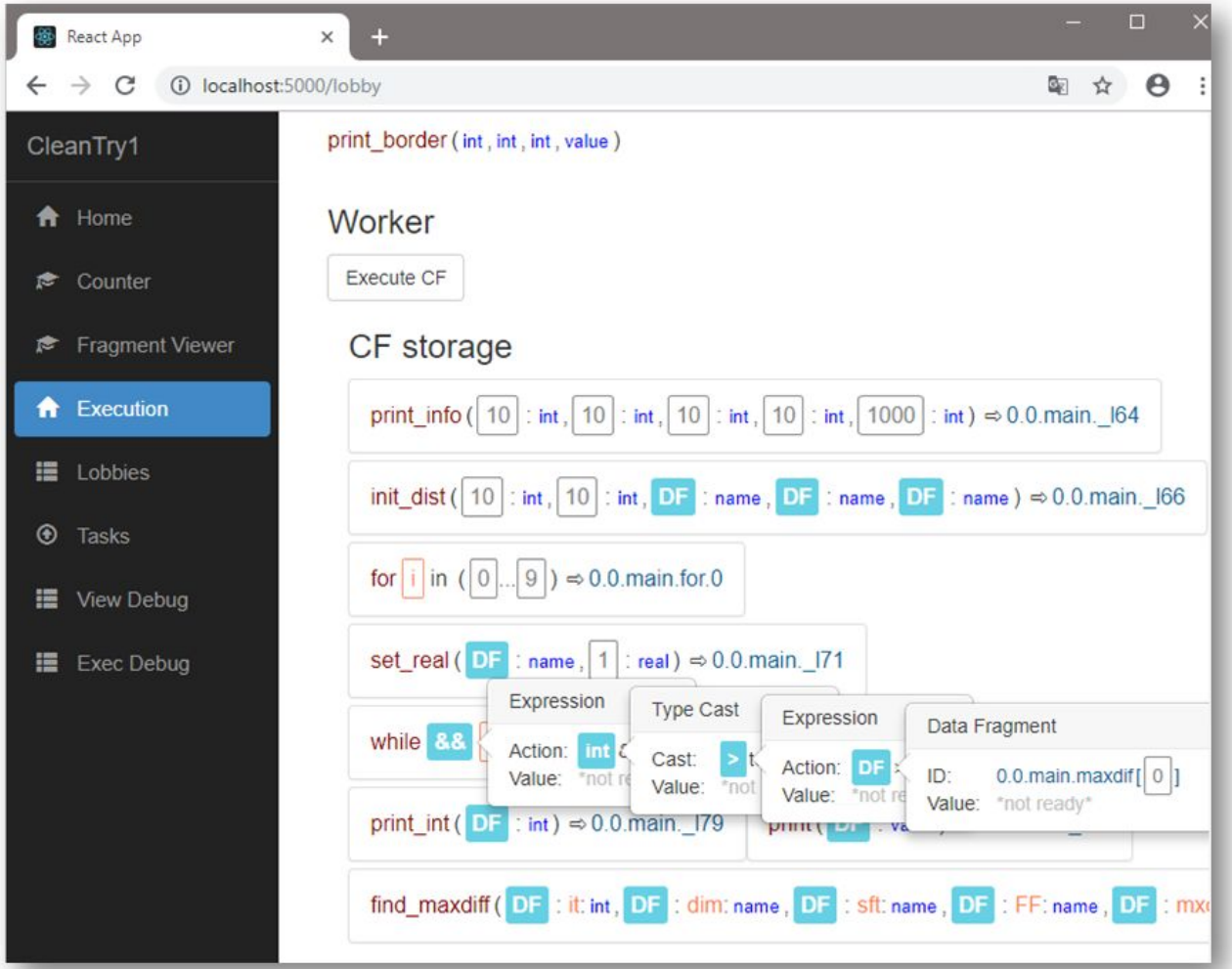
- Сформулирована модель исполнения фрагментированной программы;
- Определены функции среды исполнения;
- Разработана архитектура среды исполнения и переносимый алгоритм ее работы;
- Реализована переносимая среда исполнения;
- Выполнено тестирование, разработанная среда исполнения способна распределенно исполнять LuNA-программы и на оборудовании с низкой степенью разнородности демонстрирует поведение, аналогичное существующей среде исполнения.

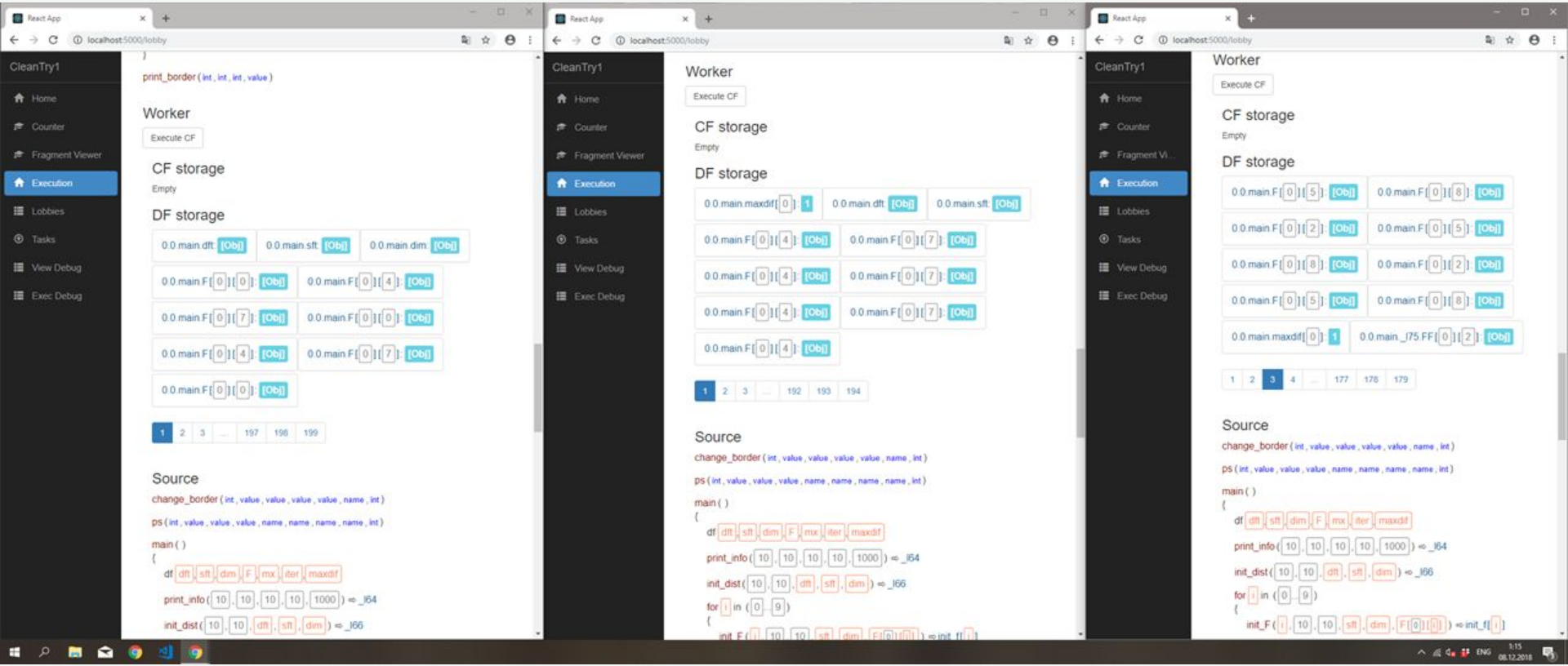
Планы:

- Провести больше тестов на неоднородном оборудовании;
- Проводить исследования системных алгоритмов исполнения параллельных программ с использованием системы.

Спасибо за внимание

Ажбаков Артем Альбертович


React App
localhost:5000/lobby
CleanTry1
Home
Counter
Fragment Viewer
Execution
Lobbies
Tasks
View Debug
Exec Debug
print_border (int , int , int , value)
Worker
Execute CF
CF storage
print_info (10 : int , 10 : int , 10 : int , 10 : int , 1000 : int) ⇨ 0.0.main._l64
init_dist (10 : int , 10 : int , DF : name , DF : name , DF : name) ⇨ 0.0.main._l66
for i in (0 ... 9) ⇨ 0.0.main.for.0
set_real (DF : name , 1 : real) ⇨ 0.0.main._l71
while &&
Expression
Action: int
Value: *not re
Type Cast
Cast: >
Value: *not
Expression
Action: DF
Value: *not re
Data Fragment
ID: 0.0.main.maxdif[0]
Value: *not ready*
print_int (DF : int) ⇨ 0.0.main._l79
find_maxdiff (DF : it: int , DF : dim: name , DF : sft: name , DF : FF: name , DF : mx


React App
localhost:5000/lobby
CleanTry1
Home
Counter
Fragment Viewer
Execution
Lobbies
Tasks
View Debug
Exec Debug
Worker
Execute CF
CF storage
Empty
DF storage
Source
change_border (int , value , value , value , value , name , int)
ps (int , value , value , value , name , name , name , name , int)
main ( )
print_info (10 , 10 , 10 , 10 , 1000) ⇒ _I64
init_dist (10 , 10 , dft , sft , dim) ⇒ _I66

Node.js vs Java

https://benchmarksgame-team.pages.debian.net/benchmarksgame/faster/javascript.html

regex-redux

source	secs	mem	gz	cpu	cpu load
Node js	12.08	850,588	408	12.78	7% 24% 33% 44%
Java	10.34	645,952	740	30.60	70% 77% 78% 73%

n-body

source	secs	mem	gz	cpu	cpu load
Node js	26.30	33,604	1297	26.30	0% 100% 3% 0%
Java	21.94	35,852	1429	21.99	0% 100% 2% 1%

mandelbrot

source	secs	mem	gz	cpu	cpu load
Node js	18.11	621,468	748	65.41	95% 87% 97% 83%
Java	5.96	79,316	796	23.34	97% 98% 98% 99%

reverse-complement

source	secs	mem	gz	cpu	cpu load
Node js	17.02	1,136,952	1103	18.52	14% 52% 35% 8%
Java	3.25	701,084	2183	7.42	51% 78% 46% 56%

fannkuch-redux

source	secs	mem	gz	cpu	cpu load
Node js	81.12	31,040	473	81.10	0% 1% 100% 0%
Java	14.30	34,412	1282	56.26	99% 98% 99% 97%

binary-trees

source	secs	mem	gz	cpu	cpu load
Node js	48.04	597,244	434	91.26	49% 44% 55% 43%
Java	8.33	985,604	835	27.40	89% 77% 81% 87%

k-nucleotide

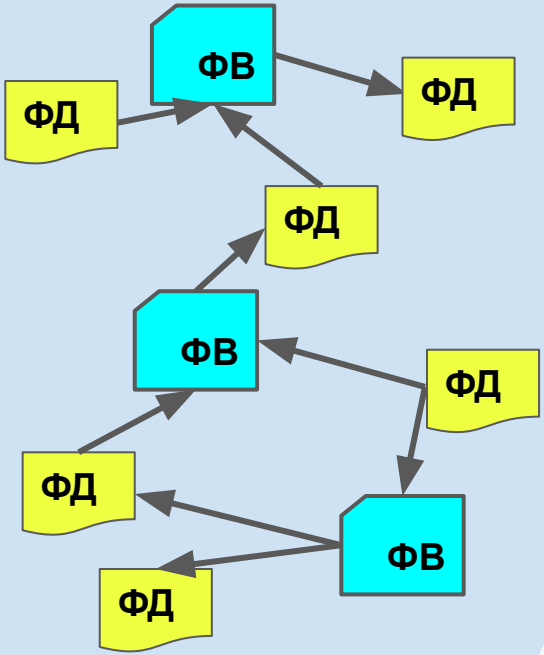
source	secs	mem	gz	cpu	cpu load
Node js	64.39	1,865,484	935	135.40	68% 66% 77% 96%
Java	8.77	465,584	1812	27.40	73% 72% 70% 98%

Заключение

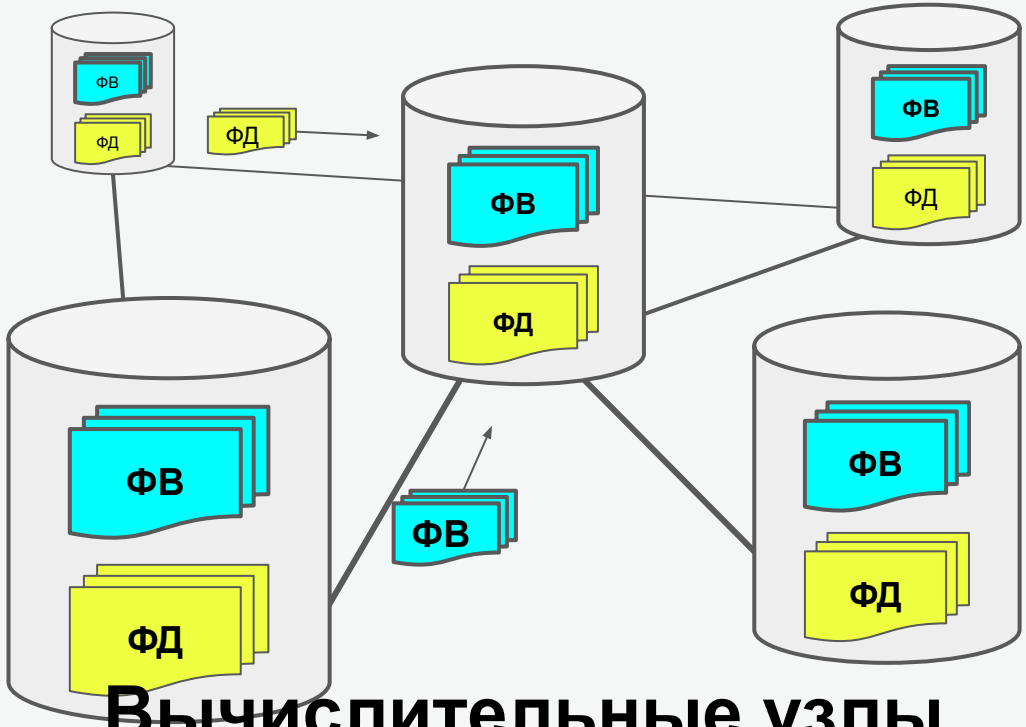
- Разработан алгоритм и архитектура системы распределенного исполнения фрагментированных программ для системы автоматизации ПП LuNA на базе web-технологий.
- Полученная система исполнения позволяет распределенно исполнять параллельные программы на разнородных вычислителях, является модифицируемой, легко развертывается на широком классе устройств.
- Произведен успешный тестовый запуск - корректное распределенное исполнение на разнородных узлах. Проведены тесты производительности.

Таким образом, разработана платформа для дальнейших экспериментов с выполнением параллельной программы в неоднородной среде.

Фрагментированное программирование



Алгоритм



Вычислительные узлы