Проект LuNA или современное системное параллельное программирование в области численного моделирования

В.А. Перепелкин

Лаборатория синтеза параллельных программ, ИВМиМГ СО РАН
Зимняя школа по параллельному программированию, 1 февраля 2016 г.
http://ssd.sscc.ru/ru/school/2016

Научное численное моделирование

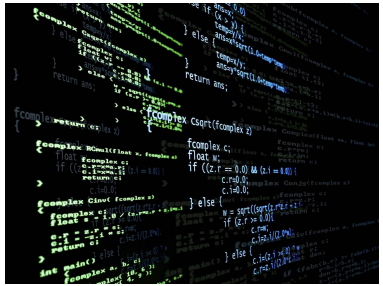
Доработка модели

$$Z(X) = c_1x_1 + c_2x_2 + \ldots + c_nx_n \to \max\ (\min)\ , \qquad (1)$$

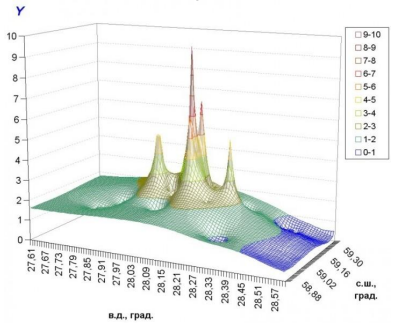
$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{in}x_n = b_i, \\ a_{(i+1)1}x_1 + a_{(i+1)2}x_2 + \ldots + a_{(i+1)n}x_n \le b_{i+1}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n \le b_m, \end{cases} \qquad (2)$$

$$x_j \ge 0,\ j = 1, 2, \ldots, t;\ t \le n. \qquad (3)$$

Математическая модель



Программа



Результаты численных экспериментов

Параллельные вычисления

Параллельные вычисления возникли практически одновременно с последовательными, но своё распространение получили позже

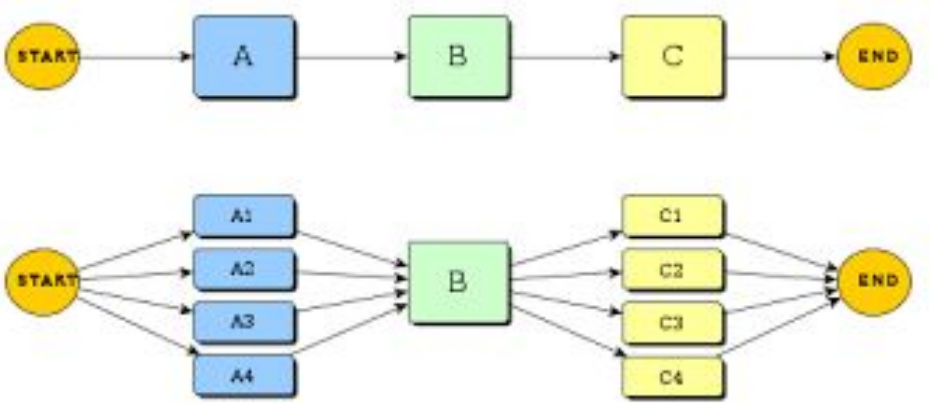
Со временем, параллельное программирование стало зачастую необходимым для численного моделирования

Возникла проблема: параллельное программирование слишком сложно

Параллельное программирование

Суть: разделить программу на части, вычисляющиеся независимо и согласованно работающие над общей задачей.

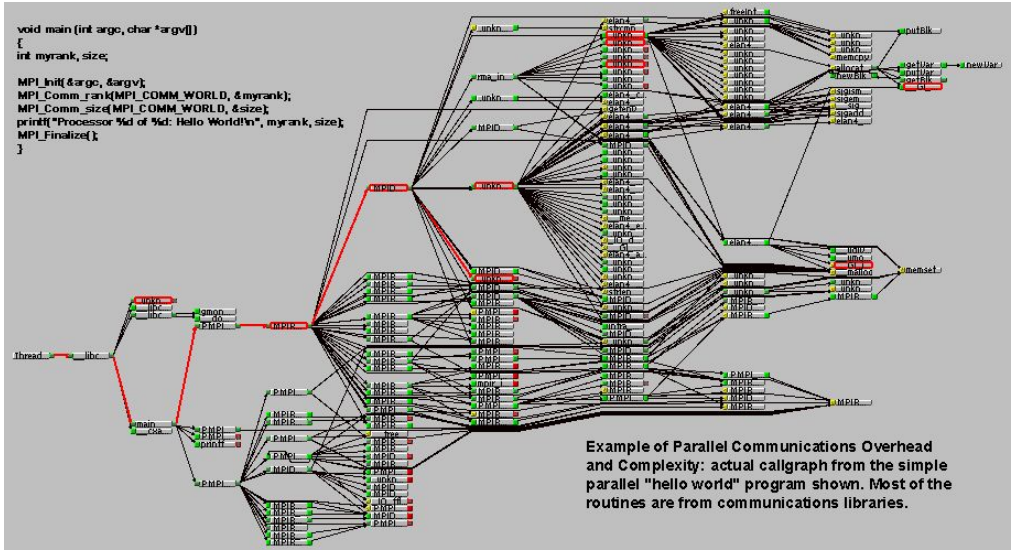
Выгоды: добавляя оборудование, мы ускоряем вычисления и увеличиваем максимальный допустимый размер задачи



Параллельное программирование

Суть: разделить программу на части, вычисляющиеся независимо и согласованно работающие над общей задачей.

Выгоды: добавляя оборудование, мы ускоряем вычисления и увеличиваем максимальный допустимый размер задачи



Проблемы параллельного программирования

- Не всегда возможно разделить задачу на части
- Даже когда декомпозиция возможна, не всегда возможно разделить задачи поровну или в течение всего времени выполнения работы
- Декомпозиция задачи — это сложная проблема, зачастую требующая творческого подхода к решению (как, например, ООП)
- Возникает дополнительная работа, связанная с организацией параллельного функционирования (например, нужно собрать частичные результаты, связь между компьютерами не мгновенная, и т.д.)
- Отладка параллельной программы гораздо сложнее вследствие *недетерминизма* и вообще наличия многих взаимодействующих процессов
- Всё это усложняет также модификацию и поддержку параллельной программы

Как результат — необходимость обращаться к профессиональным программистам

Лаборатория синтеза параллельных программ

Один из видов деятельности нашей лаборатории — помогать другим лабораториям параллельно реализовывать численные алгоритмы

Другой вид деятельности — это разработка инструментов, облегчающих разработку параллельных программ

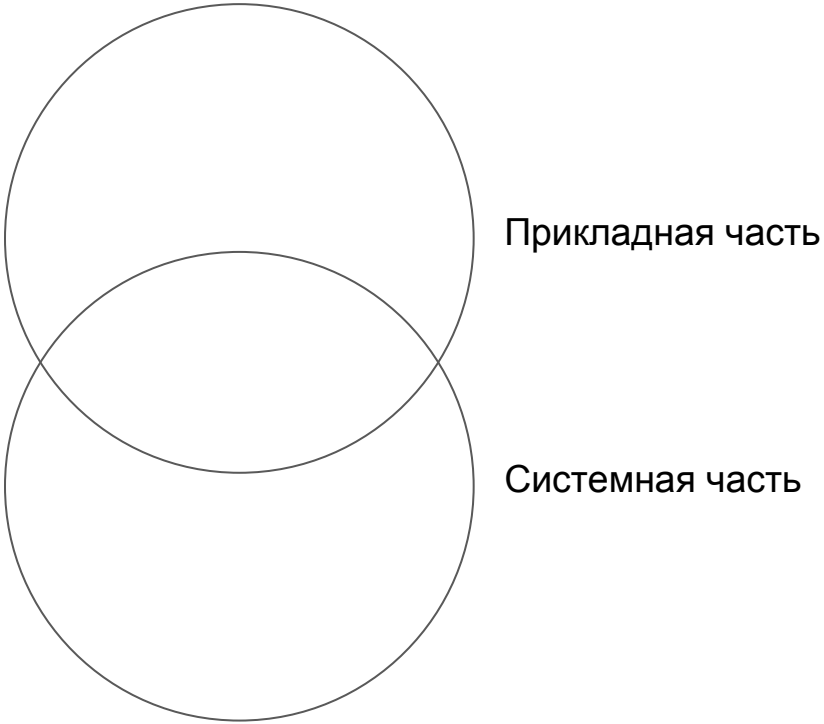
Один из таких инструментов — язык и система конструирования параллельных программ — разрабатывается в рамках проекта лаборатории: LuNA (от: **L**ang**u**age for **N**umerical **A**lgorithms)

Проект LuNA: **L**ang**u**age for **N**umerical **A**lgorithms

Цель проекта — *автоматизировать* конструирование (синтез) эффективных параллельных программ, реализующих заданный численный алгоритм для класса суперкомпьютеров

Такая автоматизация — сложная и актуальная научная проблема, которая прорабатывается мировым научным сообществом

Системное -vs- прикладное программирование



Требования к параллельным программам для суперкомпьютеров

- *Эффективность*
- *Переносимость* — на уровне сохранения эффективности
- *Масштабируемость*: сильная -vs- слабая
- Динамическая настройка на ресурсы и балансировка нагрузки на процессоры
- Локальность взаимодействий и отсутствие централизации
- Ограниченность информации о состоянии программы
- Учёт гетерогенности вычислителя
- Учёт динамики модели
- Отказоустойчивость (в перспективе)

Эффективность

- Экономия памяти
- Экономия времени вычислений
- Малая доля накладных расходов

Частичная автоматизация

Эффективная реализация численного алгоритма — алгоритмически труднорешаемая задача

Аналог: задача о рюкзаке, задача комивояжера

Вывод: автоматизация должна быть частичной, система — специализированной

Переносимость

- Переносимость на уровне исполняемых файлов
- Переносимость на уровне исходных кодов
- Переносимость с точки зрения сохранения нефункциональных свойств (эффективности)

Масштабируемость

Виды масштабируемости: сильная и слабая

Требования масштабируемости:

- Децентрализация по данным, вычислениям и коммуникациям
- Локальность коммуникаций

Отсутствует возможность иметь информацию о состоянии мультикомпьютера — сразу или всю

Идея подхода, применяемого в проекте

- Решение задачи описывается на уровне алгоритма
- Модель (представление) алгоритма *ориентирована* на автоматизацию конструирования параллельных программ
- Параллельная программа, реализующая алгоритм, конструируется автоматически
- Для обеспечения динамических свойств используется исполнительная run-time система
- Для преодоления труднорешаемых задач применяется два подхода:
  - Сужение предметной области
  - Использование высокоуровневых подсказок человека

Представление алгоритма

Разные представления алгоритма обладают разными свойствами:

- явность параллелизма
- гранулярность алгоритма
- автономность частей алгоритма
- управление ресурсами

Для разных целей удобны разные представления алгоритма

Разные уровни описания решения задачи

Формулировка задачи

Алгоритм решения

Параллельная программа

Формулировка задачи

- Функциональные требования, т.е. функция, которая должна быть вычислена
  - Пример: сортировка массива
- Нефункциональные требования (эффективность)

Алгоритм решения задачи

- Алгоритм задаёт способ вычисления функции.
- Для вычисления одной и той же функции существует счётное множество алгоритмов
- Разные алгоритмы обладают разными нефункциональными свойствами

Программа, реализующая алгоритм

Один и тот же алгоритм можно запрограммировать разными программами

Программа:

- реализует алгоритм
- управляет ресурсами
- содержит управление
- менее переносима

Структура системы LuNA

Язык описания алгоритма

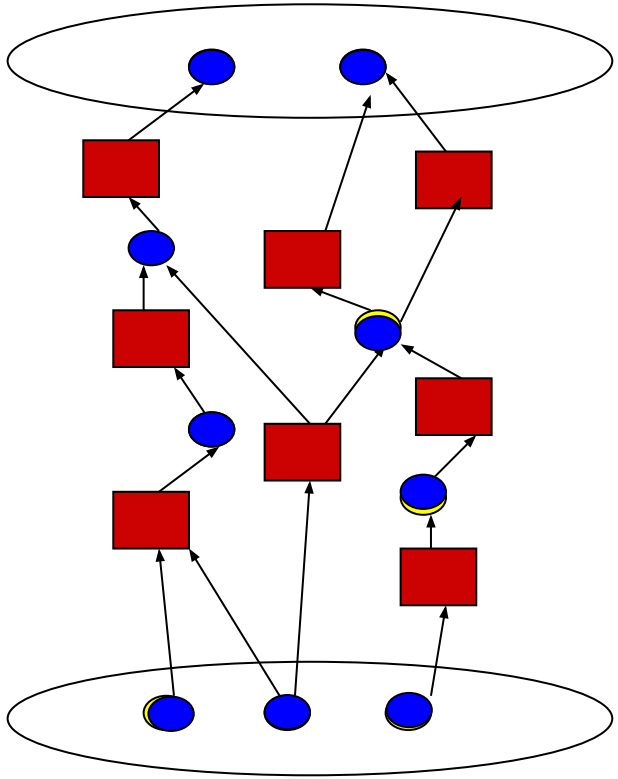
Алгоритм представляется в явно-параллельной форме, ориентированной на автоматизацию обеспечения нефункциональных свойств

Нужна, вообще говоря, другая модель описания алгоритма, обладающая нужными свойствами

Фрагментированный алгоритм (ФА)

ФА — это набор фрагментов данных (ФД), фрагментов вычислений (ФВ), и отношения in/out

Выходные ФД вычисляются из входных пока все ФВ не окажутся исполненными

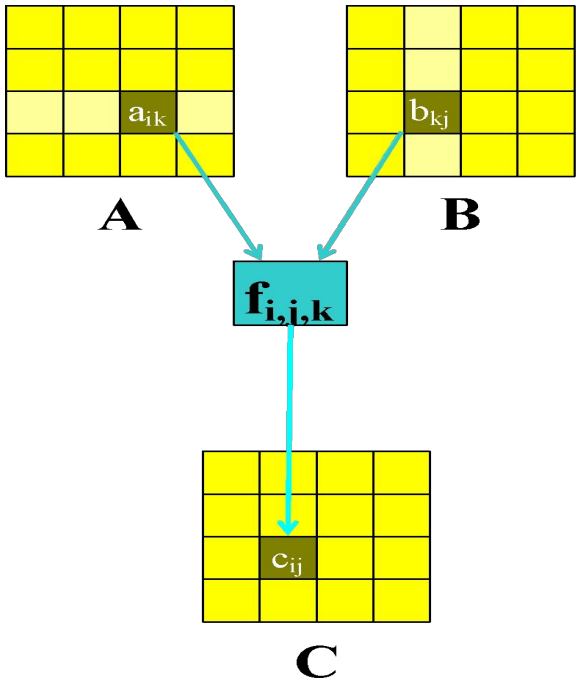


Особенности ФА

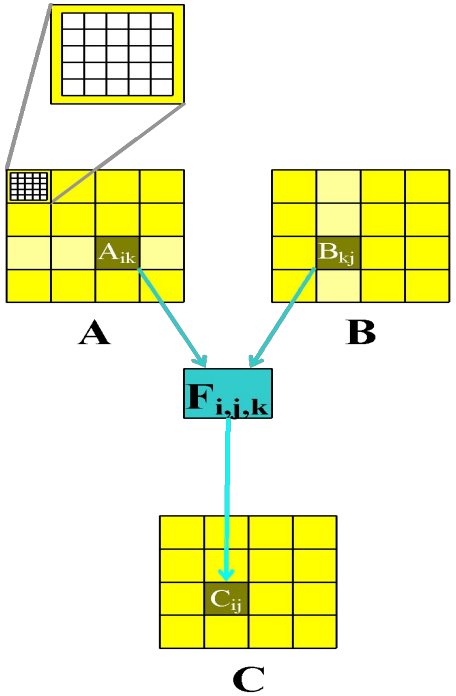
- Сериализуемые ограниченные по размеру ФД
- Ограниченные по времени фрагменты вычислений без побочных эффектов
- Единственность присваивания ФД
- Крупная зернистость ФА

Пример ФА: умножение матриц

Исходный алгоритм



Фрагментированный алгоритм



Полезные свойства ФА

- Высокая переносимость
- Ориентация на автоматизацию исполнения
  - миграция фрагментов
  - сохранение контрольных точек
  - различное управление и распределение ресурсов
  - контролируемая гранулярность
- Масштабируемость
- Явный параллелизм

Слабые стороны ФА

- Отсутствие привязки к конкретным ресурсам
- Отсутствие императивного управления
- Единственность присваивания ФД

Пример LuNA-программы 1

basic1.fa 99 Bytes

```
1  /*
2   Hello world example.
3  */
4
5  import c_helloworld() as hello_world;
6
7  sub main()
8  {
9          hello_world();
10 }
```

ucodes.cpp 81 Bytes

```
1  #include <cstdio>
2
3  extern "C"
4  void c_helloworld() {
5          printf("Hello world!\n");
6  }
```

Пример LuNA-программы 2

basic2.fa 243 Bytes

```
1  /*
2   Initialization of data fragments and data dependencies.
3  */
4
5  import c_init(int, name) as init;
6  import c_print(value) as print;
7  import c_iprint(int) as iprint;
8
9  sub main()
10 {
11         df x;
12
13         cf d: print(x);
14
15         cf b: init(7, x);
16
17         cf a: iprint(x);
18 }
```

Пример LuNA-программы 2

ucodes.cpp 362 Bytes

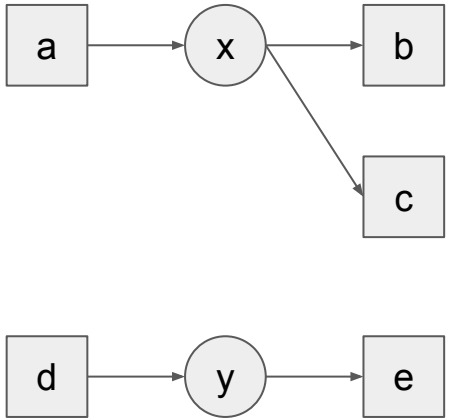
```cpp
#include <cstdio>
#include "ucenv/ucenv.h"

extern "C" {

void c_init(int val, OutputDF &df) {
        df.setValue(val);
        printf("c_init: %d --> %s, size: %d\n", val, df.getCName(), (int)df.getSize());
}

void c_iprint(int val) {
        printf("c_iprint %d\n", val);
}

void c_print(const InputDF &df) {
        printf("c_print %s is %d\n", df.getCName(), df.getValue<int>());
}

}
```

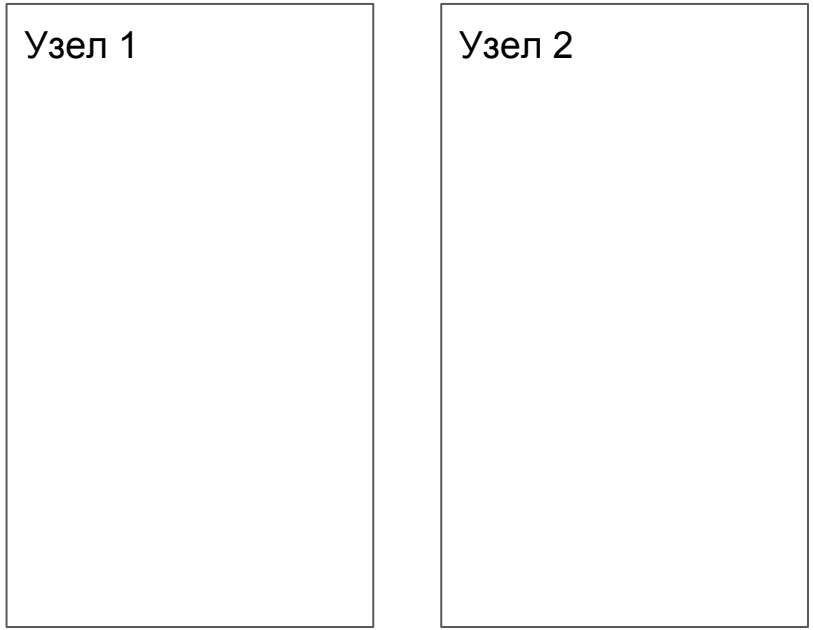
Компилятор и генератор

Задача компилятора: проанализировать алгоритм, выявить его свойства, оптимизировать алгоритм и принять частичные решения о способе его исполнения

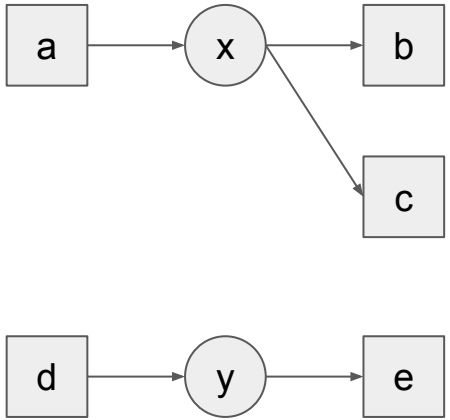
Задача генератора: по результатам анализа в компиляторе сгенерировать программу под конкретную конфигурацию вычислителя (и, возможно, под конкретные данные)

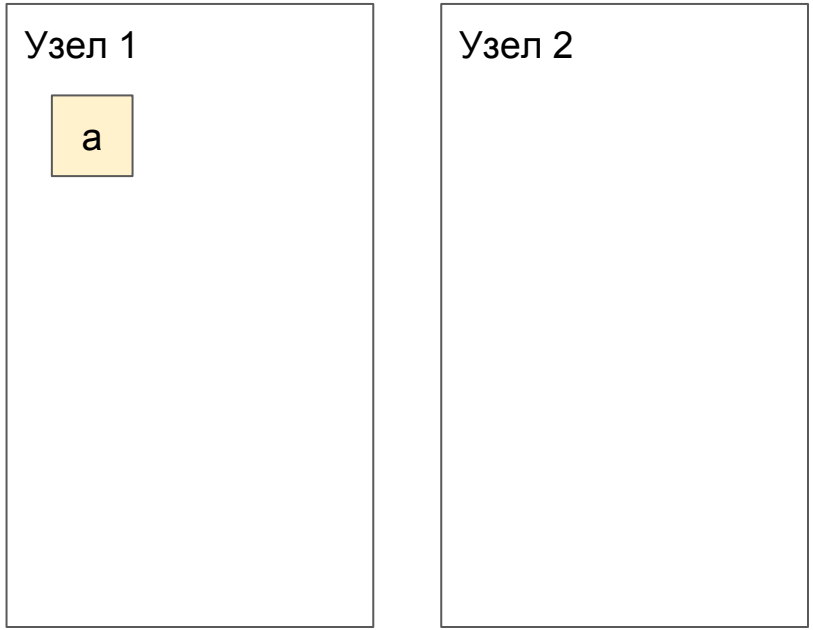
Исполнительная система: исполнение ФА



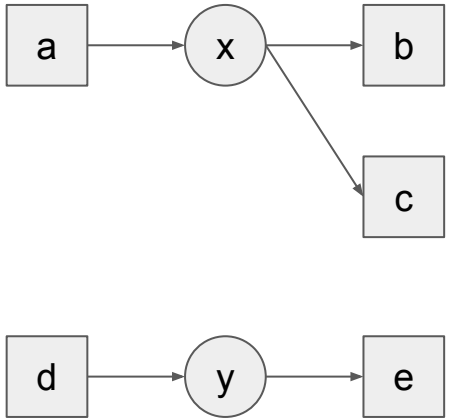


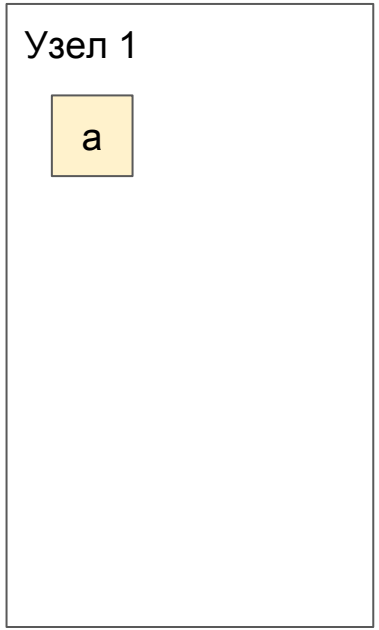
Исполнительная система: исполнение ФА

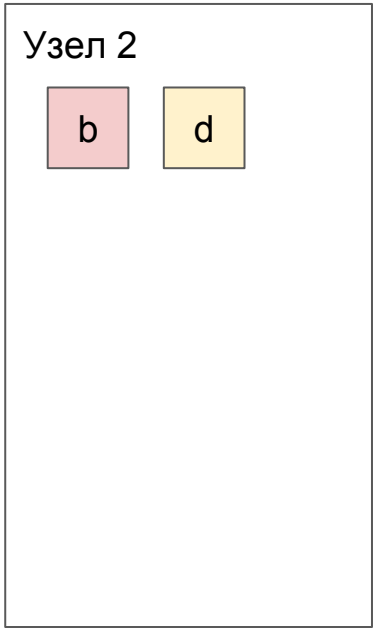




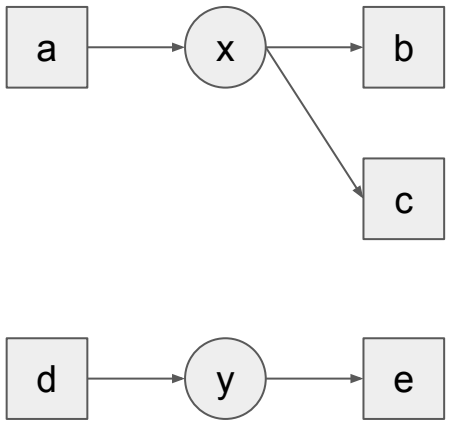
Исполнительная система: исполнение ФА

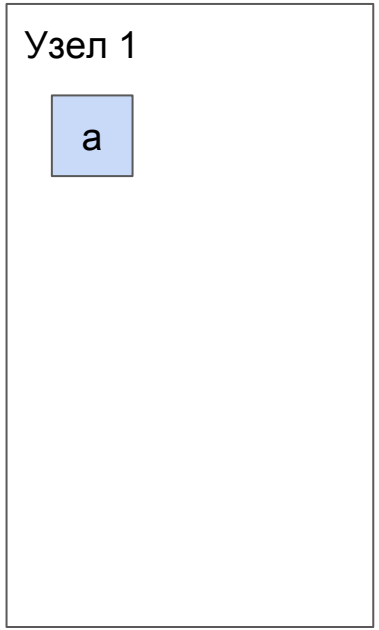


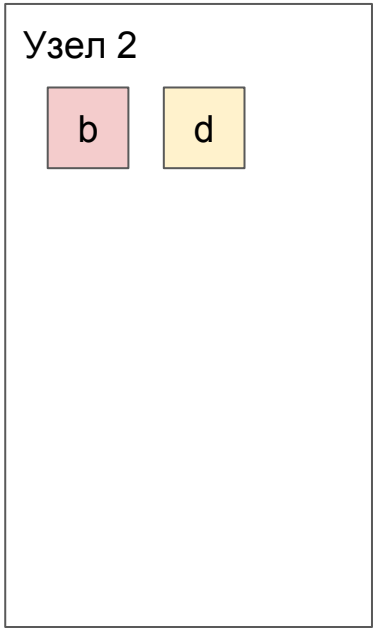




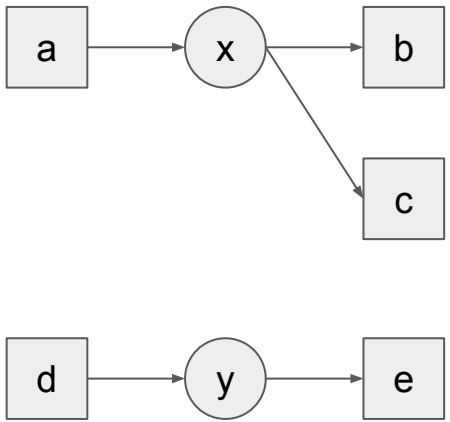
Исполнительная система: исполнение ФА

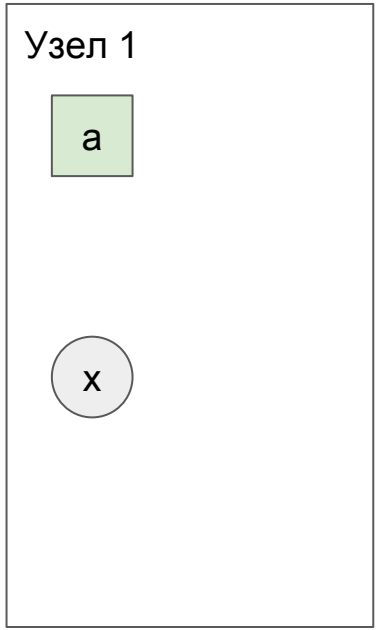


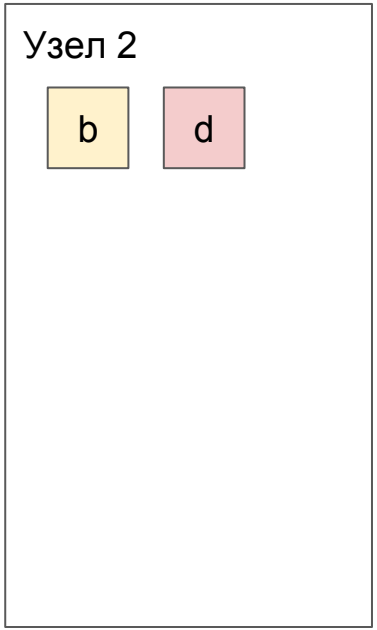




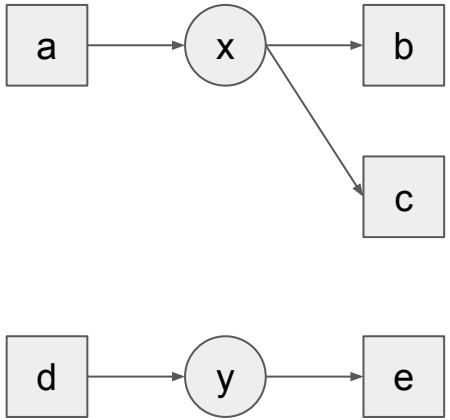
Исполнительная система: исполнение ФА

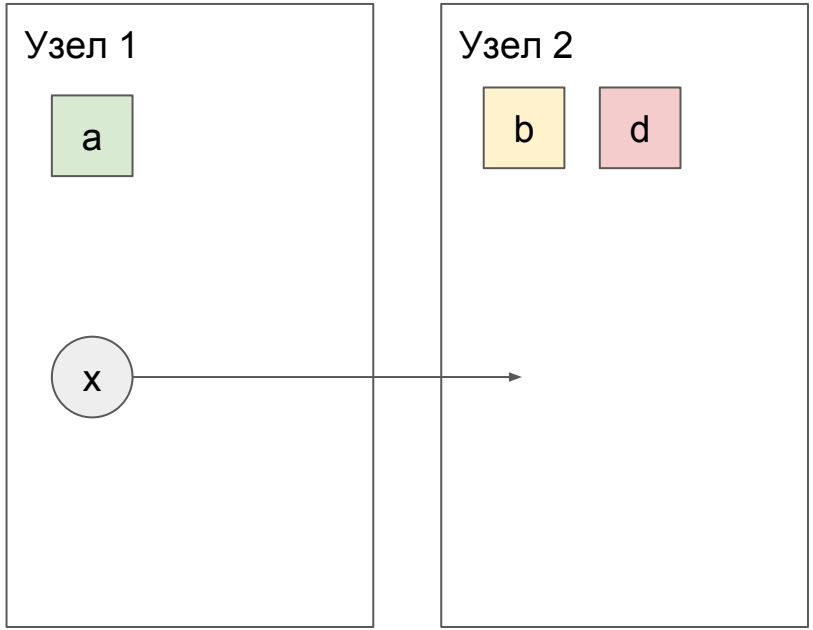




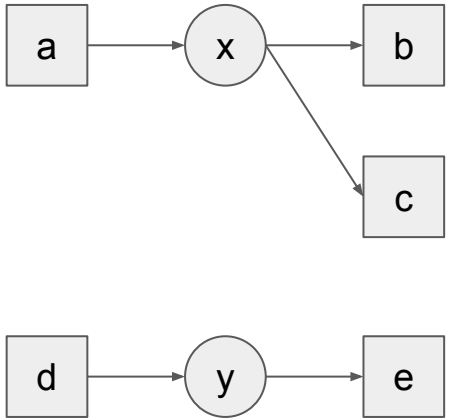


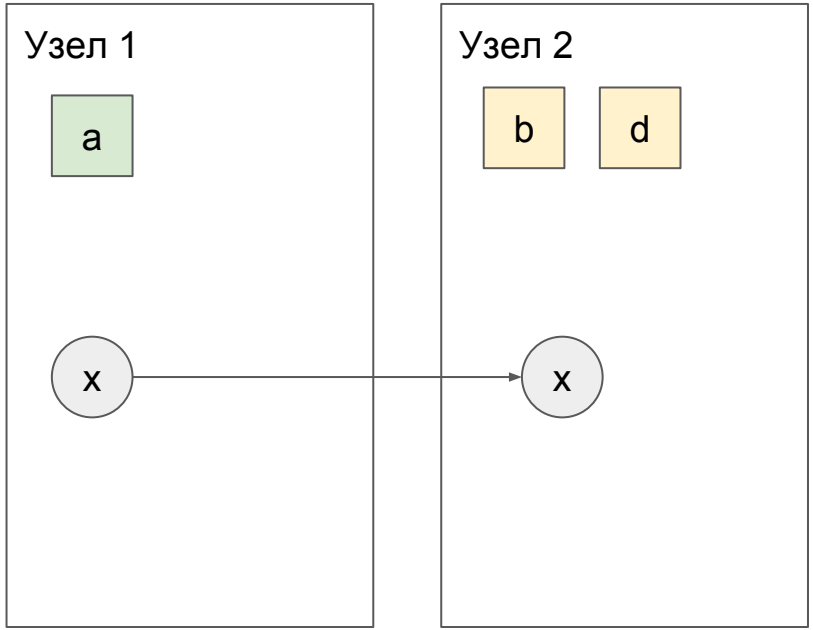
Исполнительная система: исполнение ФА



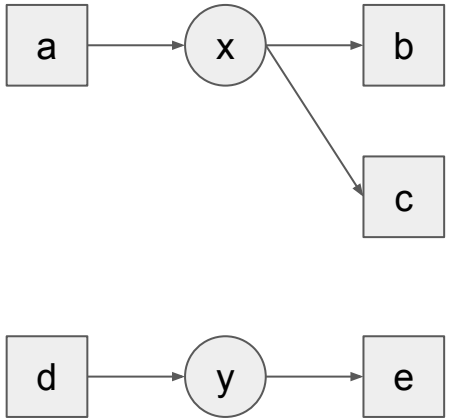


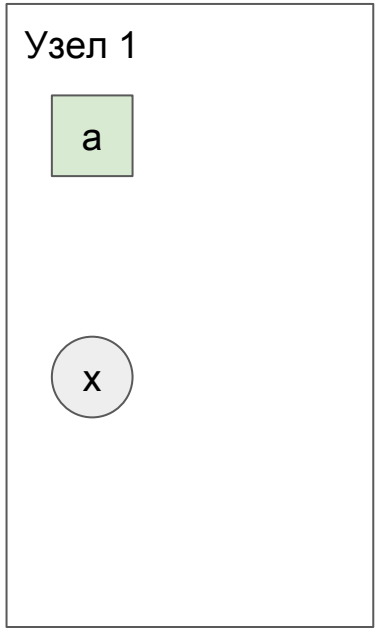
Исполнительная система: исполнение ФА

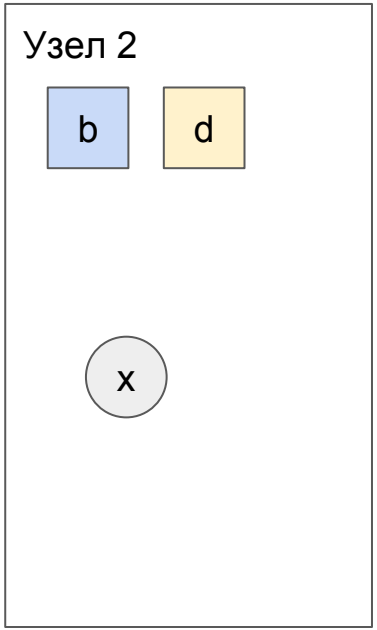




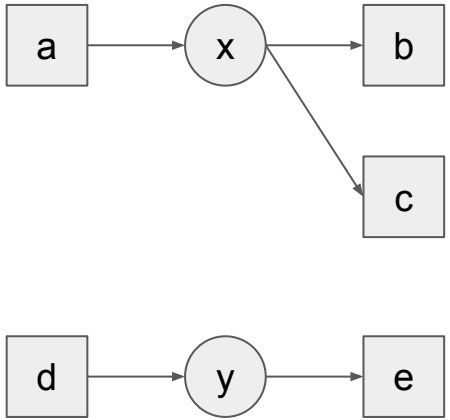
Исполнительная система: исполнение ФА

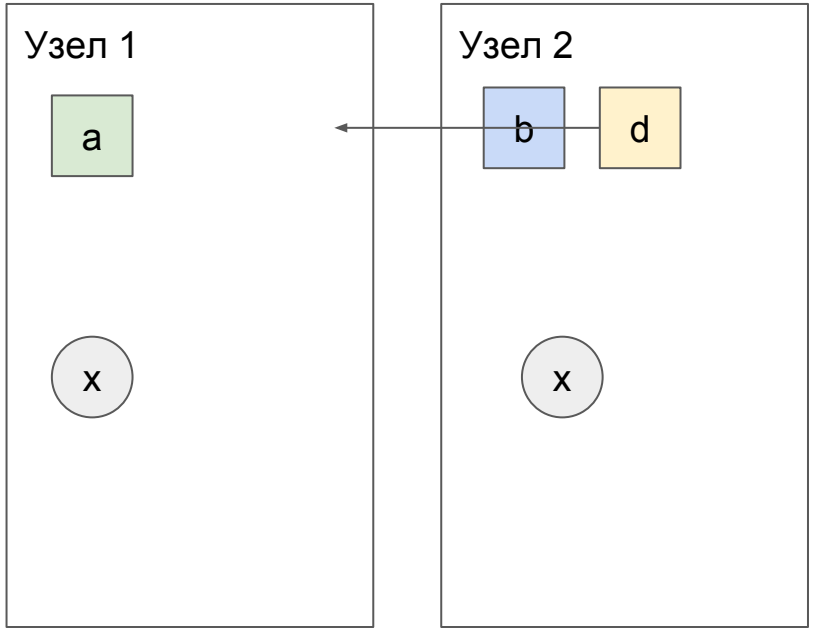




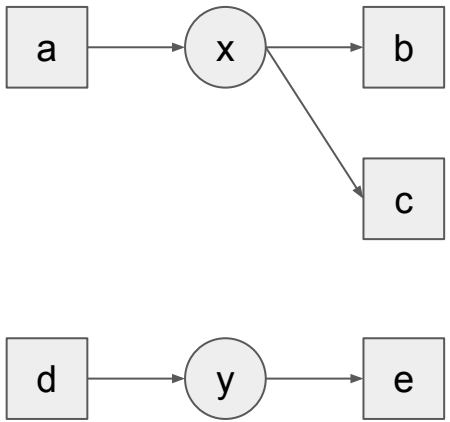


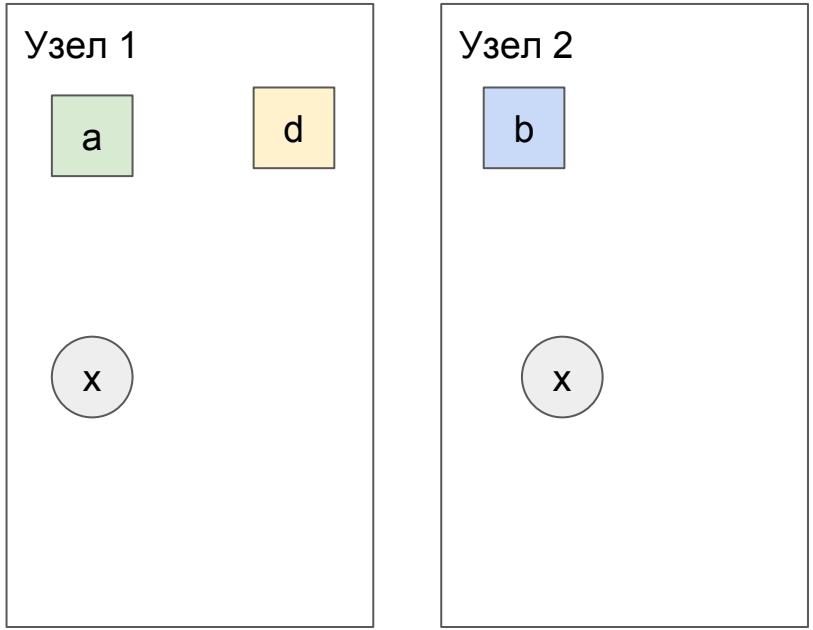
Исполнительная система: исполнение ФА



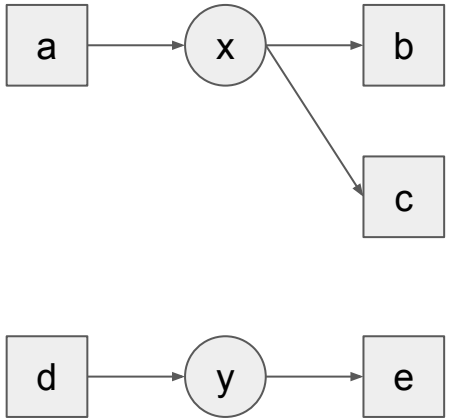


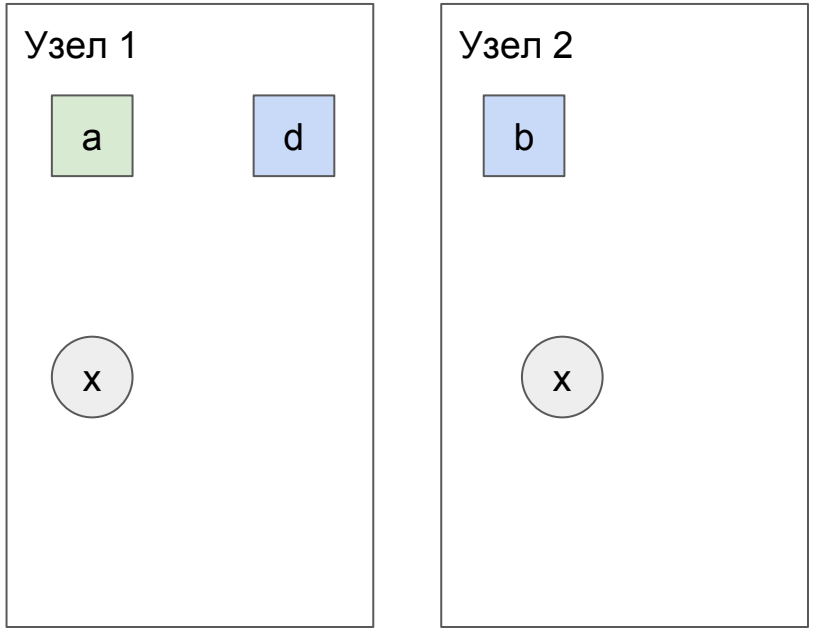
Исполнительная система: исполнение ФА



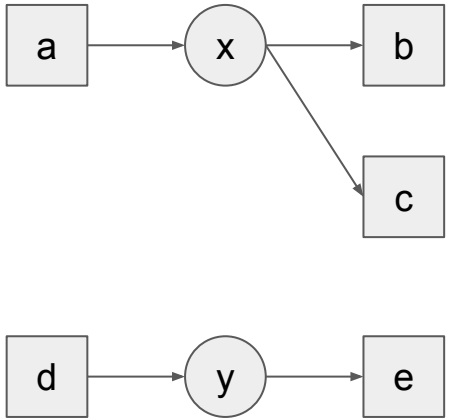


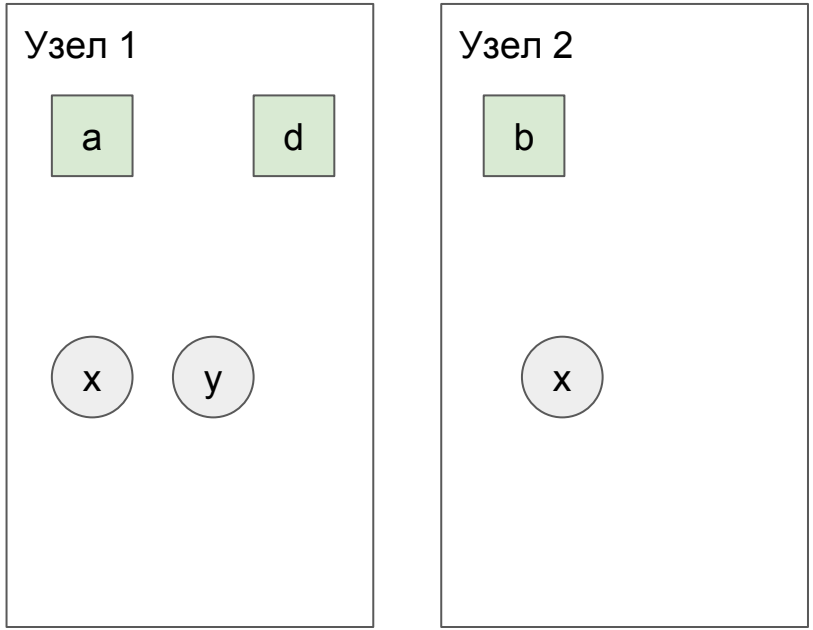
Исполнительная система: исполнение ФА



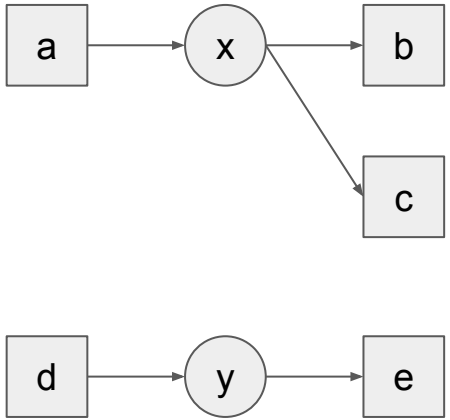


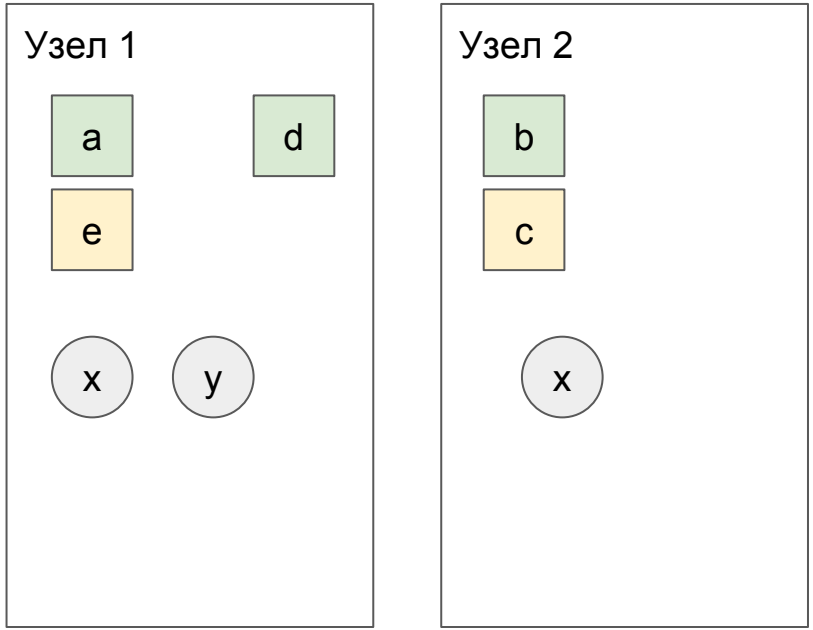
Исполнительная система: исполнение ФА



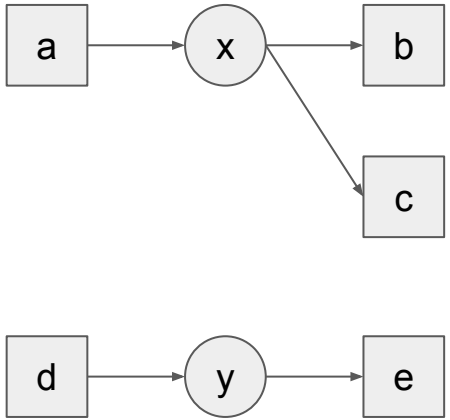


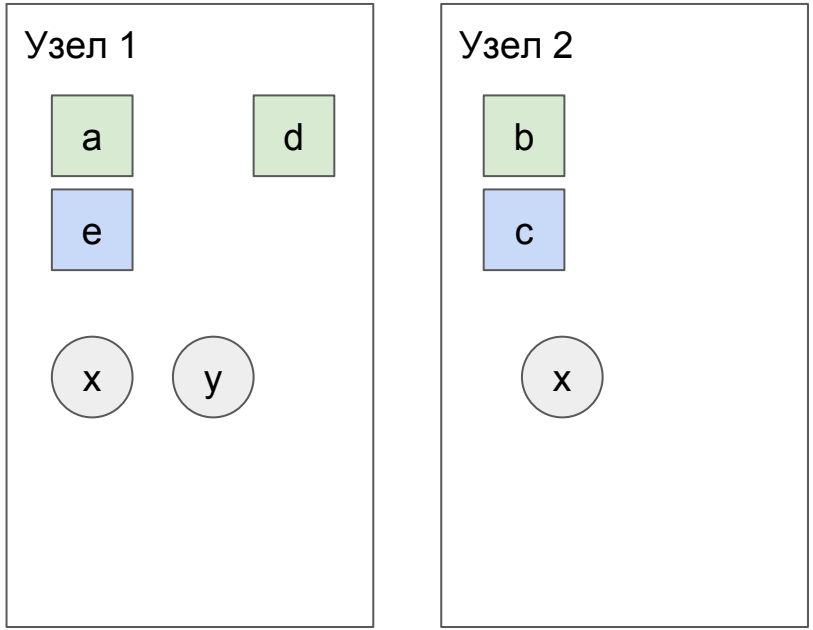
Исполнительная система: исполнение ФА



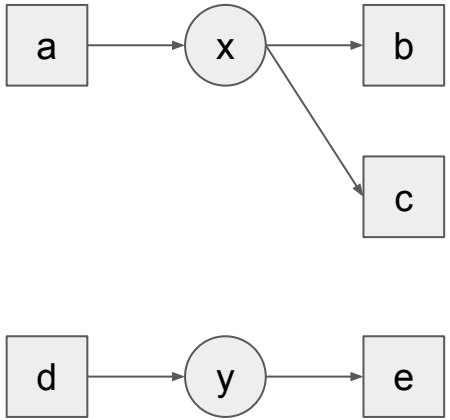


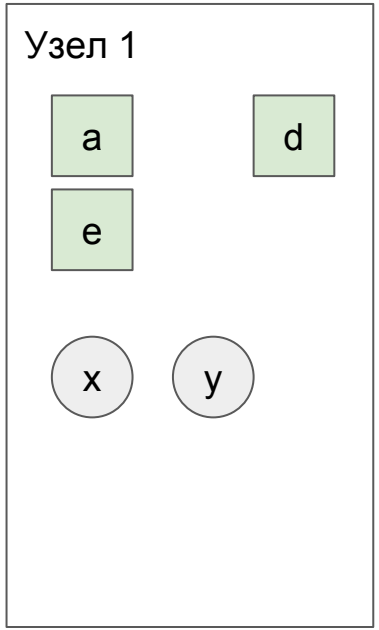
Исполнительная система: исполнение ФА

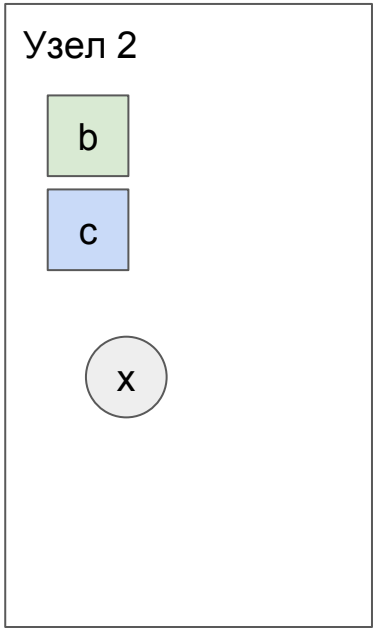




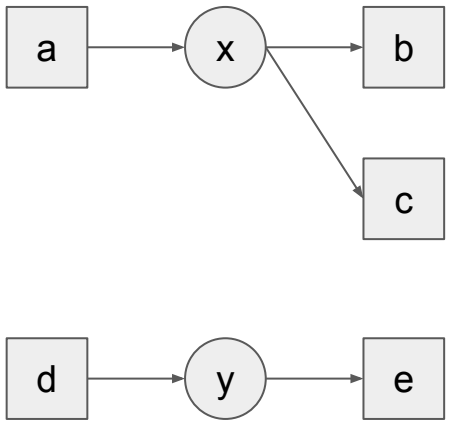
Исполнительная система: исполнение ФА

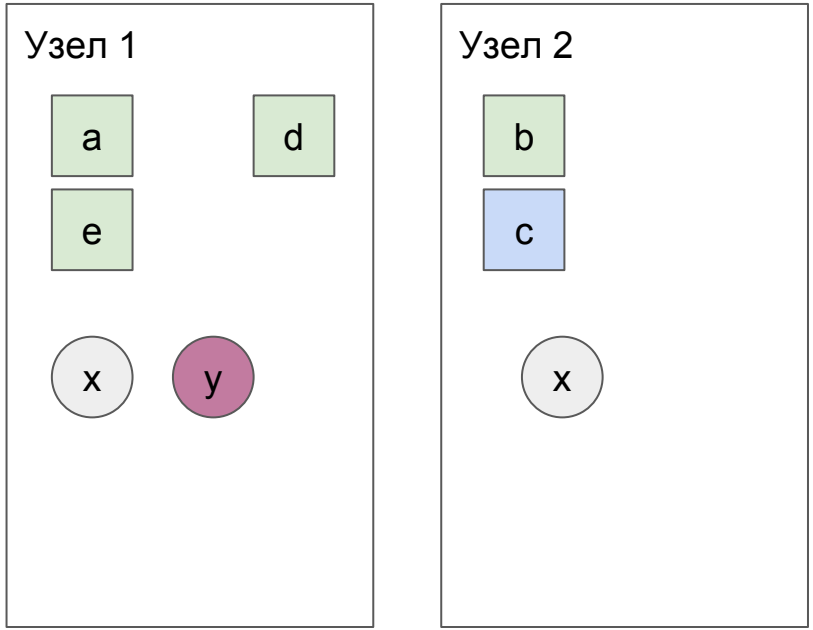




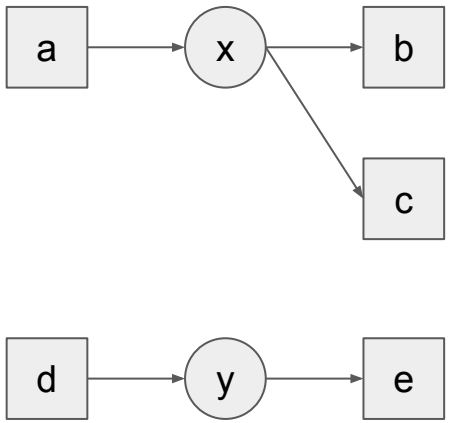


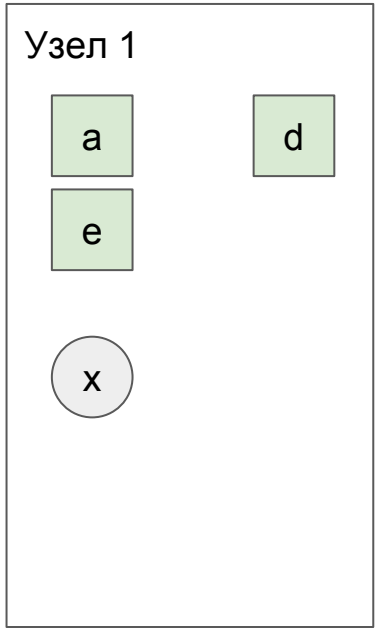
Исполнительная система: исполнение ФА

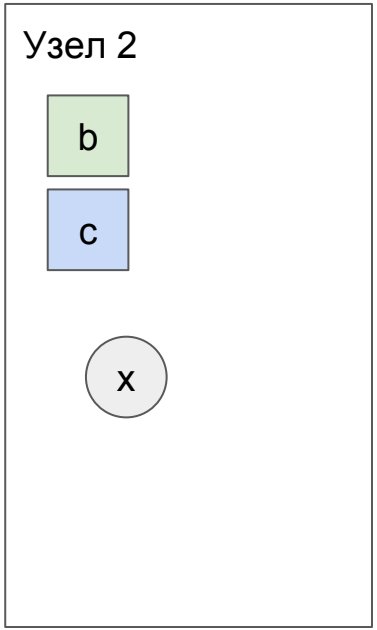




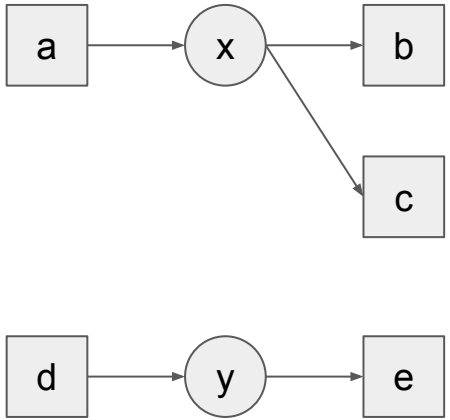
Исполнительная система: исполнение ФА

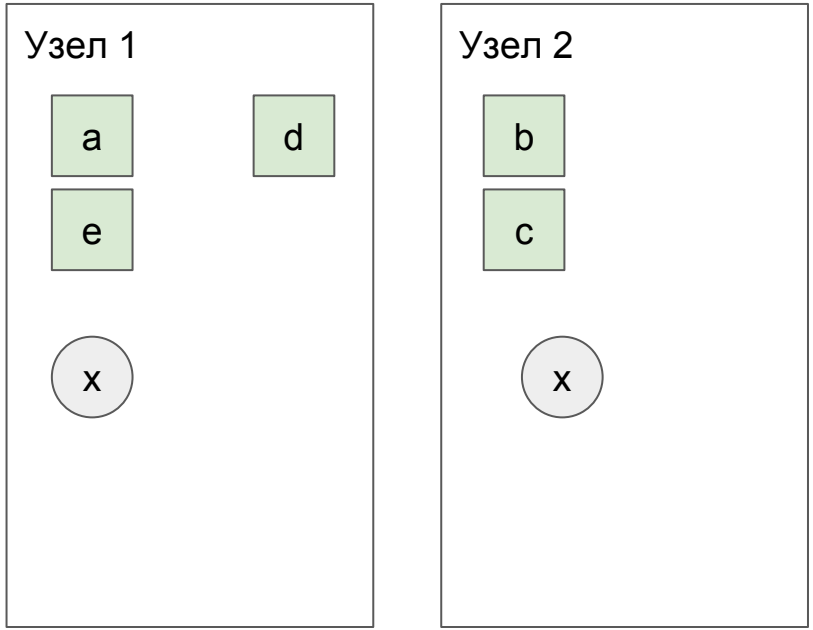




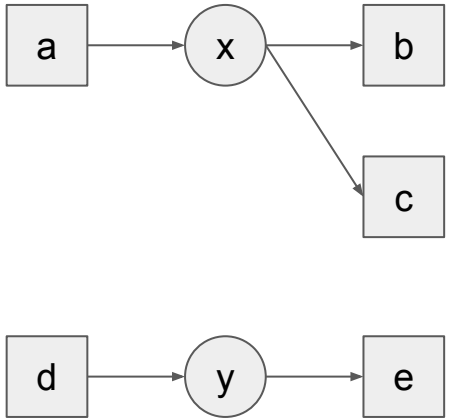


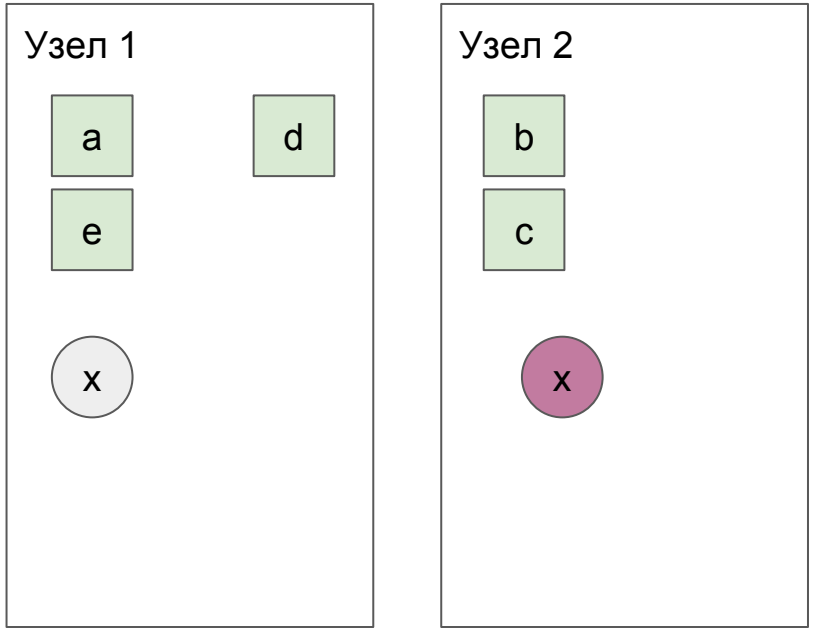
Исполнительная система: исполнение ФА



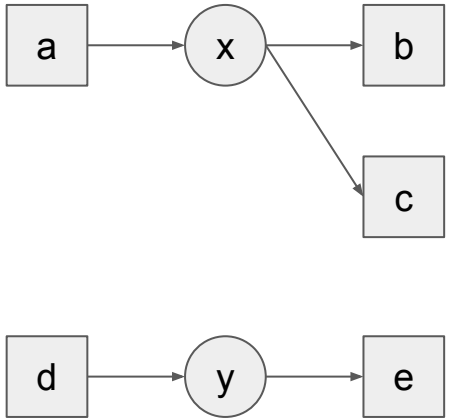


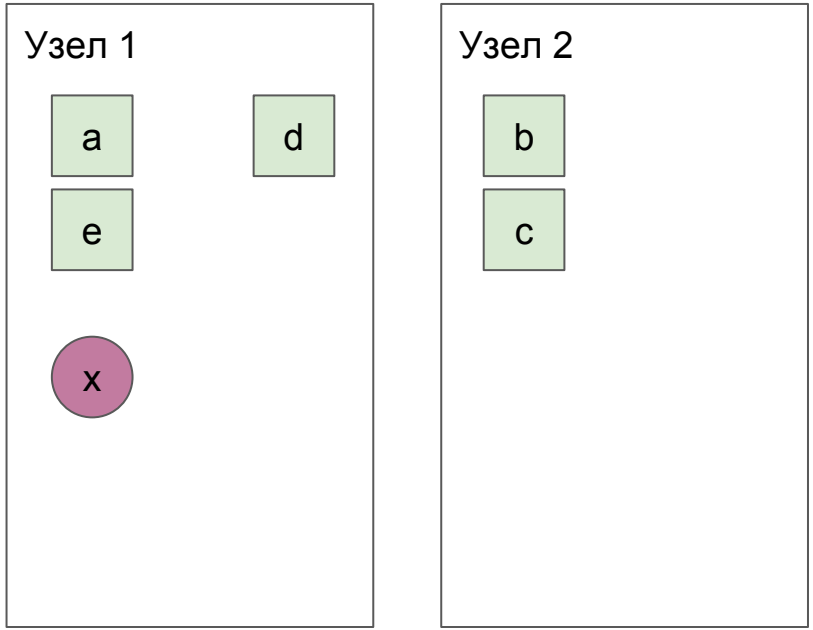
Исполнительная система: исполнение ФА



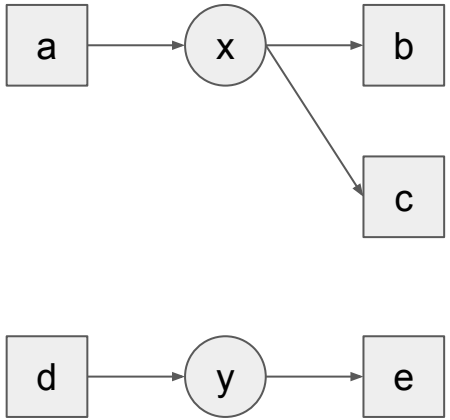


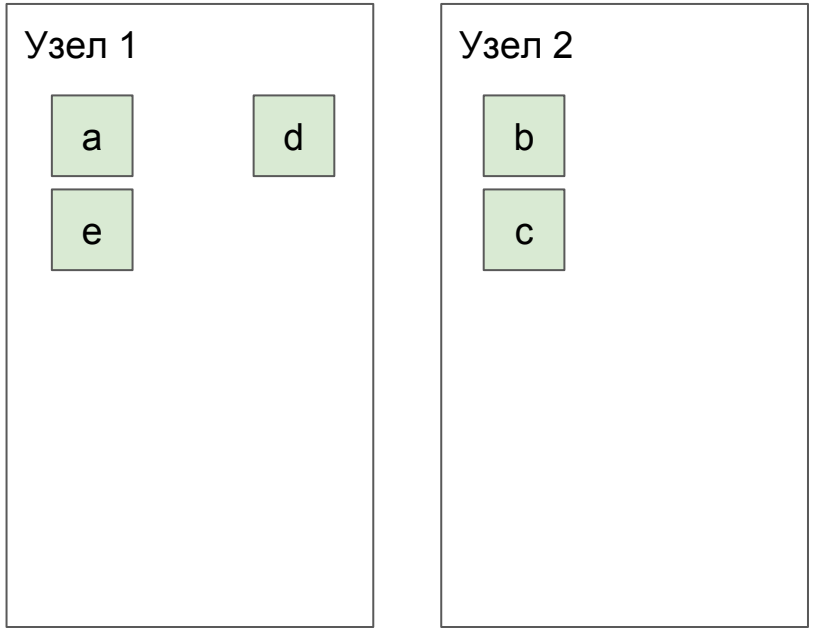
Исполнительная система: исполнение ФА





Исполнительная система: исполнение ФА





Задачи исполнительной системы

- Выполнить ФА в соответствии с информационными зависимостями
- Распределить ФД и ФВ по вычислительным узлам
- Доставить ФД на вход ФВ
- Обеспечить равномерную нагрузку на вычислитель
- Реализовать требуемые динамические свойства

Профилировщик

Задача: трассировка выполнения ФА, анализ трасс и выявление свойств исполнения ФА на имеющейся конфигурации вычислителя и входных данных

Результаты профилирования используются для оптимизации работы других компонентов системы — компилятора, генератора и исполнительной системы

Текущее состояние проекта LuNA

Имеется рабочий прототип системы со своим языком (LuNA), параллельная програма конструируется, настройка на ресурсы осуществляется автоматически, есть динамическая балансировка нагрузки на вычислительные узлы

Текущая задача: достичь хорошей эффективности (сравнимой с эффективностью программ, разработанных вручную)

Накопление прикладных программ: имеется 3 крупных приложения: PIC, IADE, Filter2D

Дальнейшие планы

Подготовка публичного релиза системы

Создание библиотеки численных подпрограмм

Совершенствование языка и системных алгоритмов

Переход к созданию системы структурного синтеза алгоритмов и программ

Спасибо за внимание!