Разработка и реализация алгоритмов автоматического конструирования LuNA-программ с использованием системы SAPFOR

Синюков Валерий Константинович
Новосибирский государственный университет
09.03.01 Информатика и вычислительная техника.
Программная инженерия и компьютерные науки
Руководитель ВКР: Малышкин Виктор Эммануилович, д.т.н., профессор, зав. каф. ПВ ФИТ
Соруководитель ВКР: Перепёлкин Владислав Александрович, к.т.н., ст. преп. каф. ПВ ФИТ
24 июня 2024 года

Введение

Для разработки параллельных программ специалистам приходится решать **сложные**, напрямую не связанные с реализуемым алгоритмом, задачи:

- Декомпозиция данных
- Обеспечение коммуникаций между процессами и потоками
- Синхронизация потоков и процессов
- Балансировка нагрузки по потокам и процессам

Параллельные программы **сложны** в отладке.

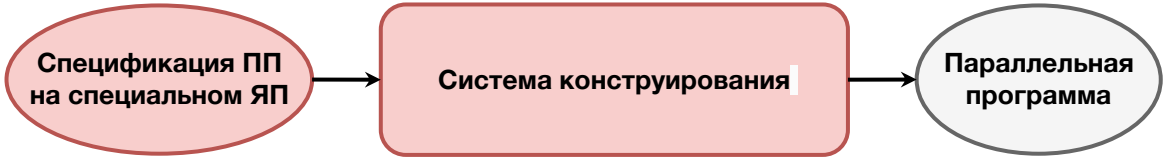
Таким образом, специалистам приходится выполнять **трудоемкую работу**, которая находится **вне области их специализации**.

Одним из подходов к решению этой проблемы является **автоматическое конструирование параллельных программ**.

Подходы к автоматическому конструированию параллельных программ

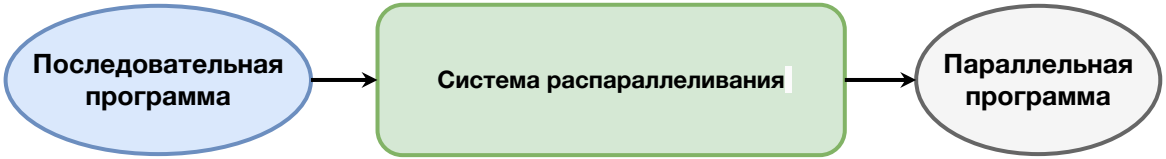
Системы, которые автоматически конструируют параллельные программы (ПП), исходя из заданной спецификации на специальном языке программирования (ЯП):

- HOPMA
- DVM
- LuNA



Системы автоматического распараллеливания последовательных программ:

- Par4All
- PLUTO
- AutoPar



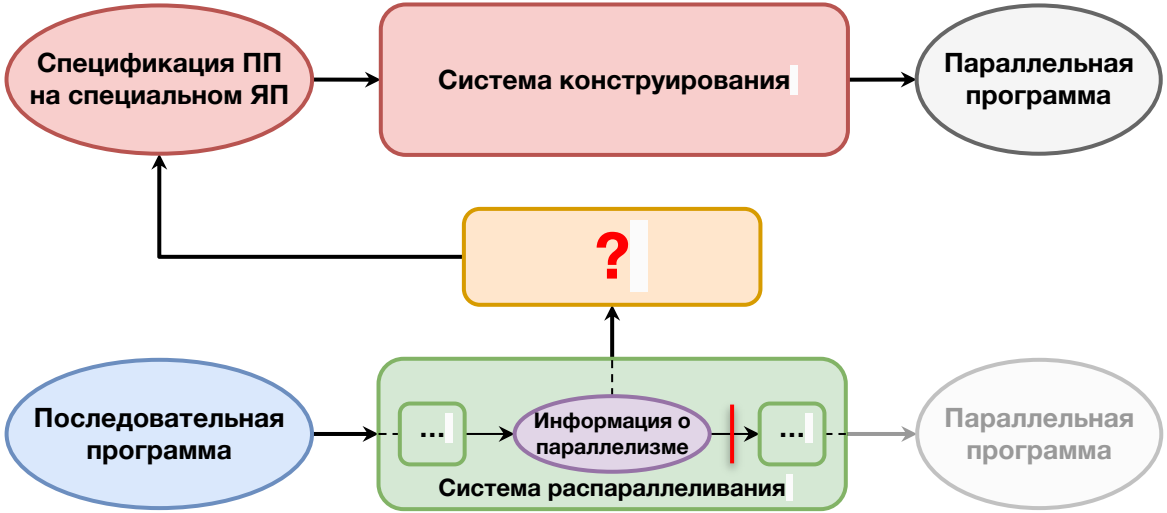
Такие системы далее будут называться **системами конструирования** и **распараллеливания** соответственно.

Связывание систем автоматического распараллеливания и автоматического конструирования

Процесс работы системы распараллеливания можно условно разделить на два этапа.

1. Извлечение из программы **информации о параллелизме**, т.е. информации о ее независимых частях.

2. Использование этой информации для **перестроения программы** таким образом, чтобы эти независимые части можно было выполнять **параллельно**.

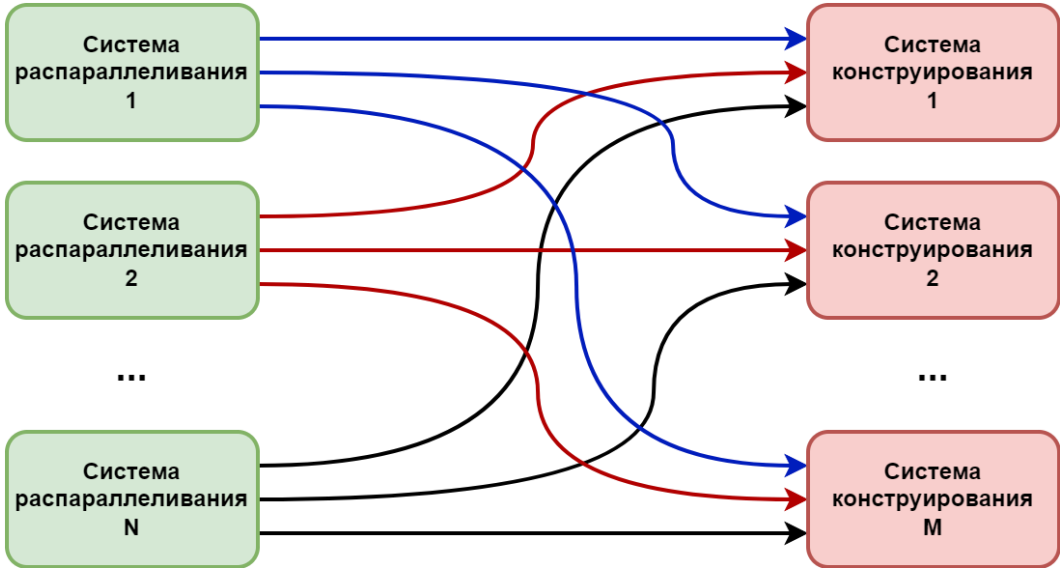


Информацию о параллелизме предлагается использовать для **автоматического создания спецификации** для системы конструирования.

Связывание систем автоматического распараллеливания и автоматического конструирования. Актуальность

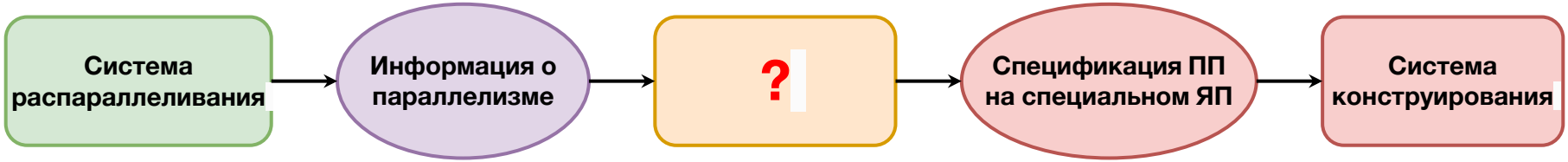
У каждой системы автоматического конструирования параллельных программ есть **своя область применения**.

Обеспечение возможности автоматического создания спецификаций для систем конструирования **расширит множество приложений**, для которых возможно автоматическое конструирование эффективных параллельных программ.



Несовместимость систем распараллеливания и систем конструирования

Системы распараллеливания и системы конструирования в общем случае работают в **разных моделях вычислений**.

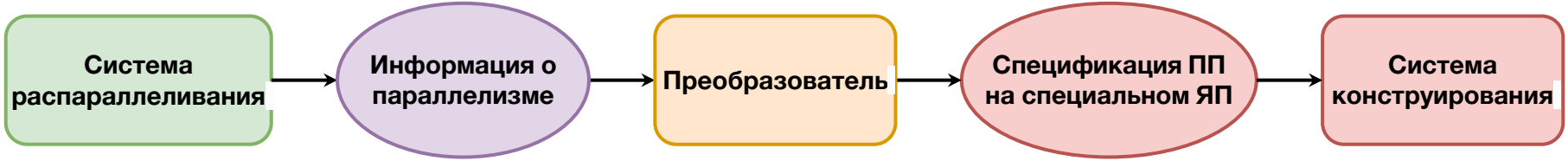


Несовместимость систем распараллеливания и систем конструирования

Системы распараллеливания и системы конструирования в общем случае работают в **разных моделях вычислений**.

Информацию о параллелизме, выявленную системой распараллеливания, нужно **дополнить** и **преобразовать**, чтобы система конструирования смогла исходя из нее создать параллельную программу.

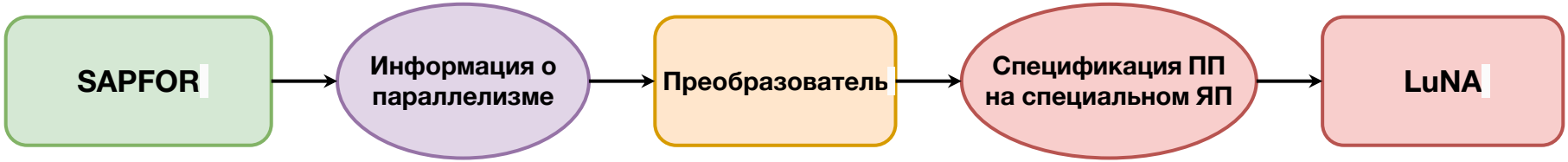
Для выполнения этой задачи требуется разработать систему, которая далее будет называться **преобразователем**.



Цель

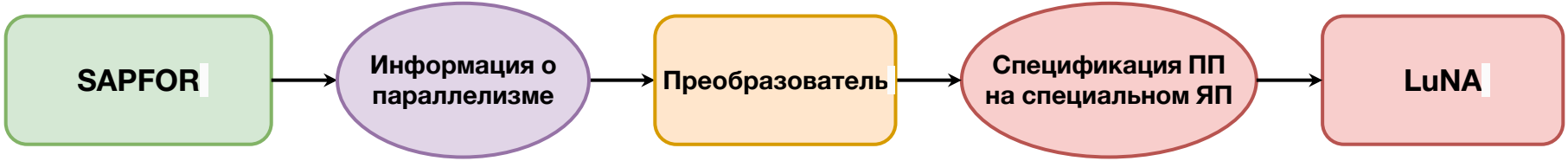
Связывание системы SAPFOR и LuNA для автоматического распараллеливания последовательных программ для класса задач на примере итерационного метода Якоби решения систем линейных уравнений.

* В рамках работы рассматриваются последовательные программы на языке программирования C++.



Задачи

1. Разработка алгоритмов дополнения информации о параллелизме, выявляемой SAPFOR.
2. Разработка алгоритмов перевода информации о параллелизме из модели вычислений, используемой в SAPFOR, в модель вычислений, используемую в LuNA.
3. Разработка алгоритмов доопределения этой информации, чтобы по ней можно было сгенерировать LuNA-программу.
4. Разработка алгоритмов генерации LuNA-программ.
5. Реализация алгоритмов генерации LuNA-программ и их экспериментальное исследование на примере прикладной задачи.



Обзор

Области, в которых неизбежно возникает задача определения одного представления программы из другого:

1. Системы, основанные на **полиэдральных моделях**: PoCC, Polly, PLUTO.
2. Фреймворк **MapReduce**, с помощью которого конструируются программы для обработки больших объемов данных (1 Тб и более).
3. Распараллеливание программ на основе **директив**: OpenMP, DVM, Cilk, Cilk++.
4. Промежуточные представления компиляторов, например, **LLVM IR**.

Автору неизвестно о существовании системы или модели, которая могла бы выполнять работу преобразователя в полной мере.

Система распараллеливания SAPFOR

SAPFOR — System **FOR A**utomated **P**arallelization

SAPFOR может выполнять анализ программ на языках C++ и Fortran с целью выявления информации о параллелизме.

Работает в модели вычислений потока управления.

Разрабатывается в ИПМ им. М.В. Келдыша РАН.

http://dvm-system.org/ru/

М.С. Клинов, В.А. Крюков. Автоматическое распараллеливание Фортран-программ. Отображение на кластер // Вестник Нижегородского университета им. Н.И. Лобачевского, Нижний Новгород: Издательство Нижегородского госуниверситета, 2009, № 2, С. 128-134

Система конструирования LuNA

LuNA — Lang**u**age for **N**umerical **A**lgorithms

Система LuNA по описанию задачи в терминах предметной области строит параллельную программу.

Работает в модели вычислений фрагментированного программирования.

Разрабатывается в ИВМиМГ СО РАН.

https://ssd.sscc.ru/luna

Victor E. Malyshkin, Vladislav A. Perepelkin. LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // Parallel Computing Technologies. 11th International Conference, PaCT 2011, Proceedings. LNCS 6873. Springer, 2011. pp. 53-61.

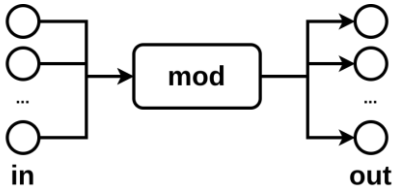
LuNA-программа

LuNA-программа задает спецификацию требуемой параллельной программы.

Данные представлены с помощью переменных **единственного** присваивания, которые называются *фрагментами данных*.

Вычисления представлены с помощью операций, которые называются *фрагментами вычислений*.

Фрагмент вычислений — это триплет вида

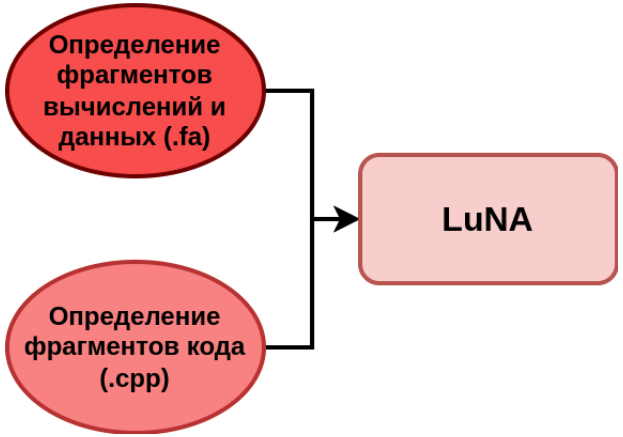


Где

- **in** — множество входных фрагментов данных.
- **out** — множество выходных фрагментов данных.
- **mod** — модуль, который из входных фрагментов данных вычисляет выходные. В LuNA модуль называется *фрагментом кода*.

Фрагменты данных и фрагменты вычислений описываются на языке программирования LuNA.

Фрагменты кода описываются на языке программирования C++.

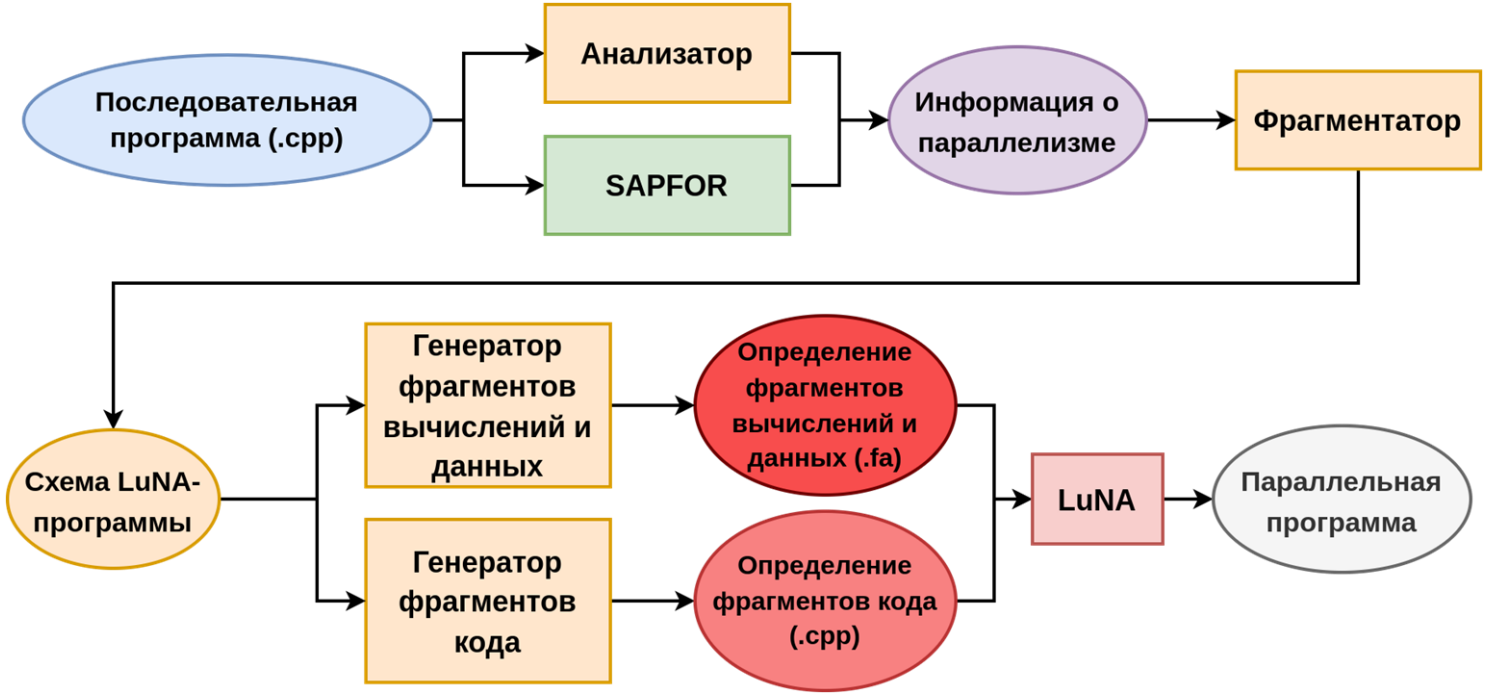


LuNA-программа

Для описания потенциально бесконечного множества фрагментов вычислений и данных в LuNA присутствует **массовые** и **условный** операторы: `for`, `while`, `if`.

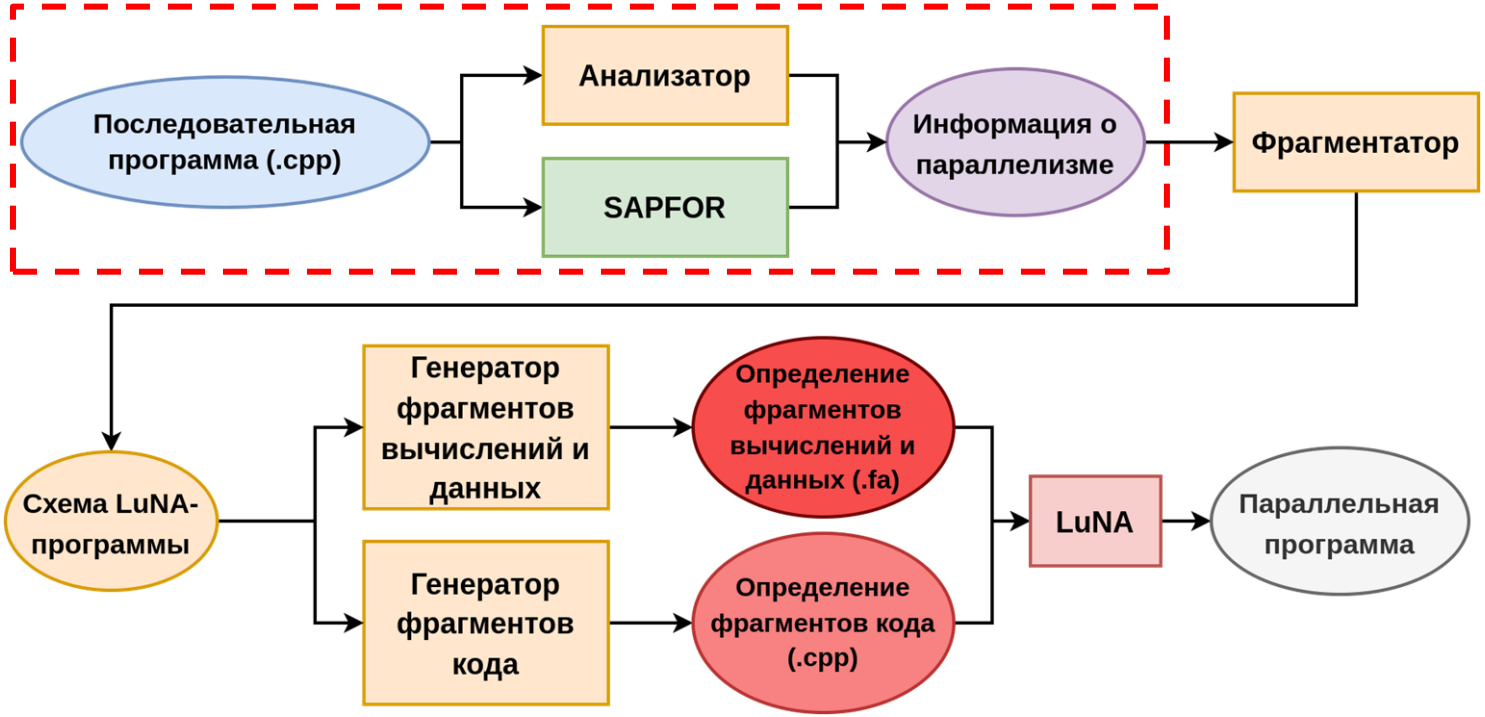
Имена фрагментов данных строятся из **базовой** и **индексной** частей.

Предлагаемый подход к автоматическому распараллеливанию последовательных программ на основе систем SAPFOR и LuNA



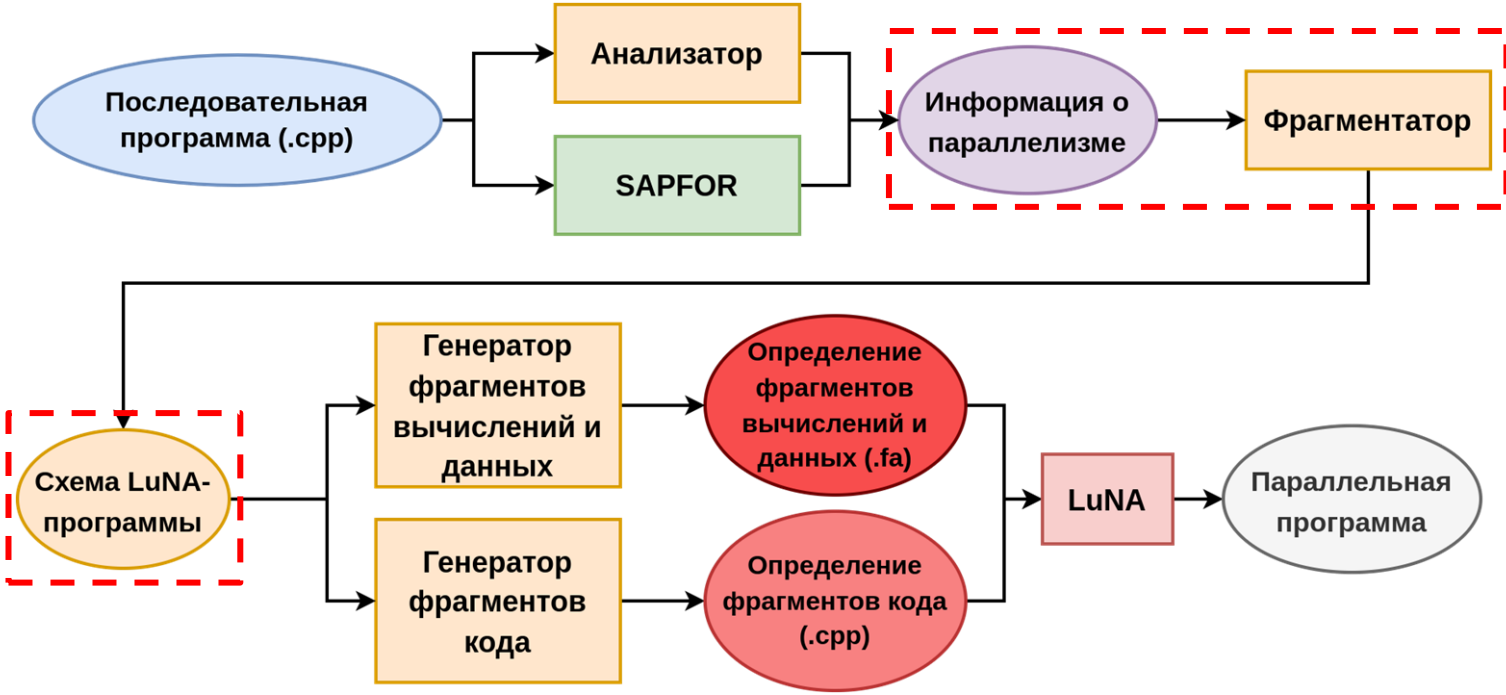
Предлагаемый подход к автоматическому распараллеливанию последовательных программ на основе систем SAPFOR и LuNA

SAPFOR и анализатор извлекают из исходной последовательной программы информацию о параллелизме.



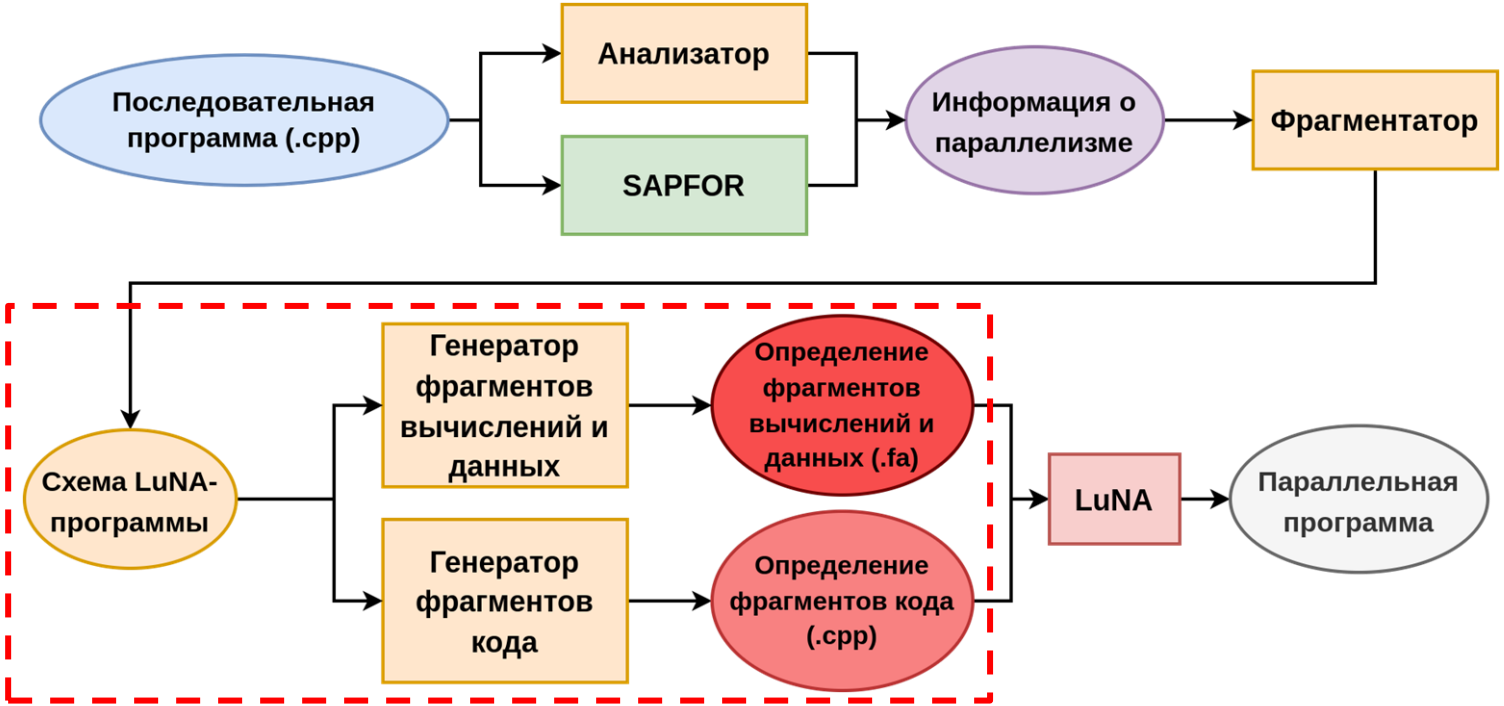
Предлагаемый подход к автоматическому распараллеливанию последовательных программ на основе систем SAPFOR и LuNA

Фрагментатор по информации о параллелизме составляет схему LuNA-программы.



Предлагаемый подход к автоматическому распараллеливанию последовательных программ на основе систем SAPFOR и LuNA

По схеме LuNA-программы генераторы фрагментов кода, вычислений и данных создают LuNA-программу.



Фрагментация по пространству — параллельное выполнение циклов

Составляется разбиение множества итераций **вычислительноемких гнезд циклов** без зависимостей.

Предлагается выполнять каждое множество из этого разбиения в **отдельном фрагменте вычислений**.

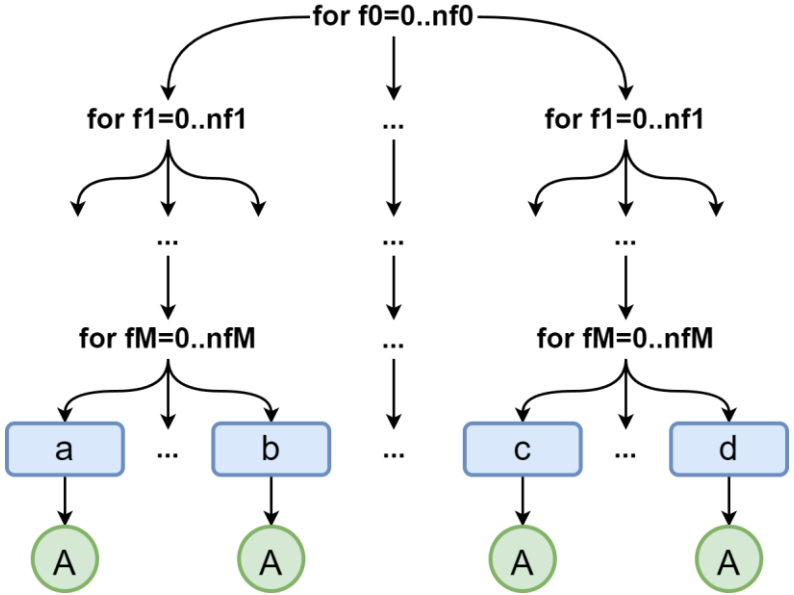
Для выполнения фрагментации по пространству нужно:

- Определить фрагменты кода исходя из преобразованных циклов исходной программы.
- Описать фрагменты вычислений с помощью гнезда массовых операторов `for`.

Исходная программа

```
01: for (int i0 = 0; i0 < N0; ++i0)
02:   for (int i1 = 0; i1 < N1; ++i1)
03:     ...
04:       for (int iM = 0; iM < NM; ++iM)
05:         // вычисление элемента массива без побочных
06:         // эффектов
07:         A[i0][i1]...[iM] = calc_no_side_effects(i0, i1, …,
08:                                                 iM);
```

LuNA-программа



Фрагментация по пространству — параллельное выполнение циклов

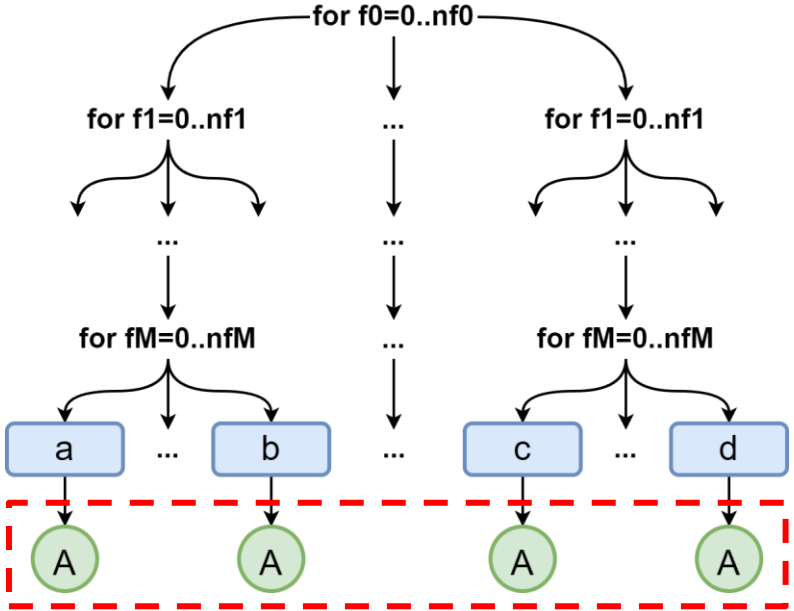
Массивы тоже могут фрагментироваться по пространству, то есть хранится ни в одном, а в **нескольких фрагментах данных**.

При фрагментации массивов по пространству может возникнуть потребность использования **теневых граней**.

Исходная программа

```
01: for (int i0 = 0; i0 < N0; ++i0)
02:   for (int i1 = 0; i1 < N1; ++i1)
03:     ...
04:       for (int iM = 0; iM < NM; ++iM)
05:         // вычисление элемента массива без побочных
06:         // эффектов
07:         A[i0][i1]...[iM] = calc_no_side_effects(i0, i1, …,
08:                                                 iM);
```

LuNA-программа



Фрагментация по времени — представление множественного присваивания

Предлагается при именовании фрагментов данных использовать **временны́е индексы**.

Временные индексы соответствуют порядку выполнения фрагментов вычислений.

Исходная программа

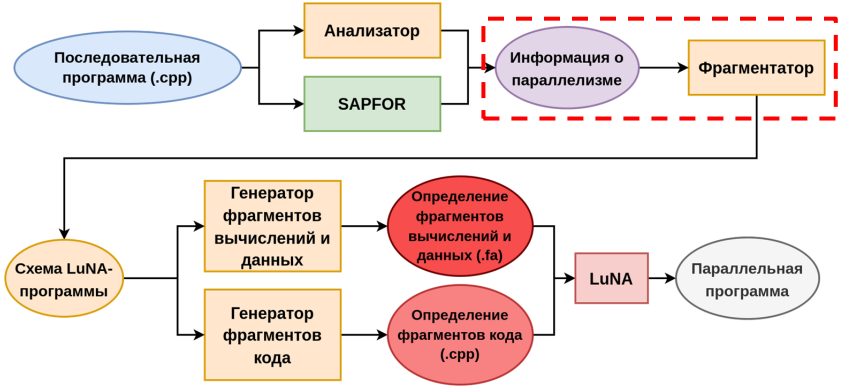
```
01:int main()
02:{
03:    int a;
04:    a = 3; // запись значения переменной a
05:    for (int i = 0; i < 50; ++i)
06:    {
07:        a *= 2; // чтение и запись значения
08:                // переменной a
09:    }
10:}
```

LuNA-программа

```
01: sub main() {
02:    df a;
03:    init_a(a[0]);
04:    destructive_consumption(a[0], a[1][-1]);
05:    for i=0..49 {
06:       loop_body(a[1][i], a[1][i-1]);
07:    }
08:    destructive_consumption(a[1][49], a[2][-1]);
09: }
```

Проблемы, которые должен решить фрагментатор

- Разбиение программы на **фрагменты кода**.
- **Декомпозиция данных**: какие переменные должны быть представлены с помощью фрагментов данных, какие — с помощью локальных переменных фрагментов кода.

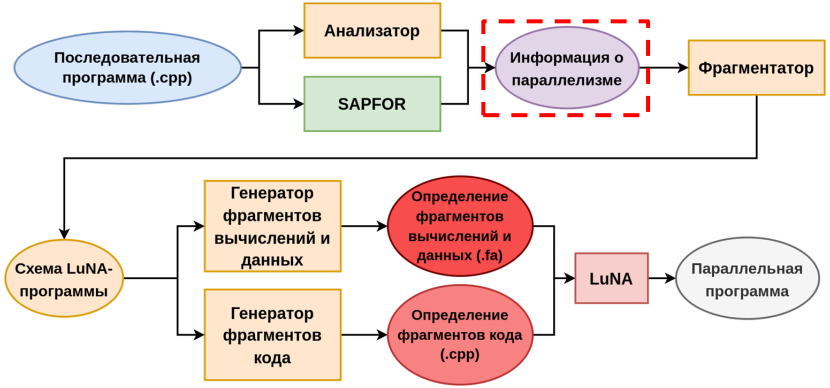


Информация о параллелизме

Информация о параллелизме достаточна для того, чтобы по ней выполнить **декомпозицию данных и вычислений**.

Информация о параллелизме содержит информацию о:

- Циклах
- Зависимостях по данным и управлению
- Редукционных операциях
- Побочных эффектах
- Вызываемых процедурах
- Используемых переменных
- Доступах к элементам массивов



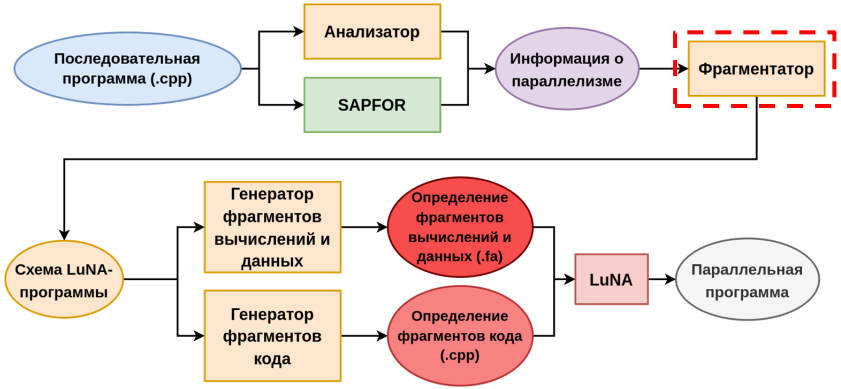
Для хранения информации о параллелизме разработан формат на основе JSON.

Фрагментатор

Задачей фрагментатора является **создание схемы LuNA-программы** исходя из информации о параллелизме.

Для этого он определяет:

- Каким образом части исходной программы определяют фрагменты кода.
- Переменные, которые должны быть представлены фрагментами данных.
- Фрагментируемые по пространству циклы и массивы.
- Циклы, которые должны быть представлены с помощью массовых операторов.
- Фрагменты вычислений, которые нужно выполнять упорядоченно, даже если это не следует из информационных зависимостей.



- Информационные зависимости, следующие из порядка выполнения операторов исходной программы.
- Фрагменты кода, в рамках которых должны использоваться и/или инициализироваться теневые грани массивов.

Схема LuNA-программы

Схема LuNA-программы содержит принципиальные решения о том, как LuNA-программа должна быть устроена, без технических деталей.

Схема LuNA-программы содержит информацию о принципиальной организации **фрагментов кода**:

- Сигнатура:
  - Имя фрагмента кода
  - Параметры (имя, тип, массив ли, теневые грани)
- Тело:
  - Вызов процедуры
  - Выделение памяти под массив
  - Доступ к элементу массива
  - Цикл
  - Последовательный участок кода

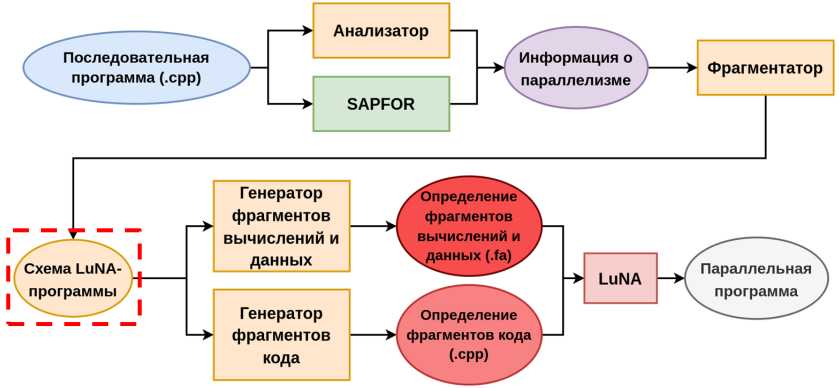


Схема LuNA-программы

Схема LuNA-программы содержит принципиальные решения о том, как LuNA-программа должна быть устроена, без технических деталей.

Схема LuNA-программы содержит информацию о принципиальной организации **фрагментов вычислений**:

- Массовый оператор:
  - Счетчик
  - Фрагментация по времени
  - Упорядоченность фрагментов вычислений
  - Условия прерывания
- Применение фрагмента кода:
  - Имя фрагмента кода
  - Аргументы (значение, тип, редукционные операции, входной ли, выходной ли, фрагментация по времени)
  - Упорядоченность

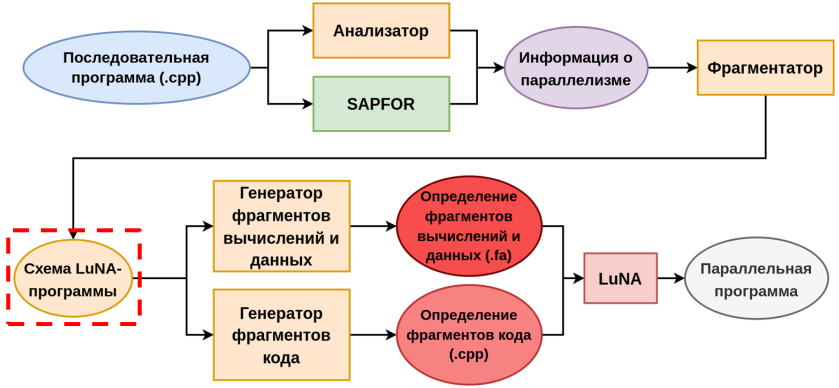
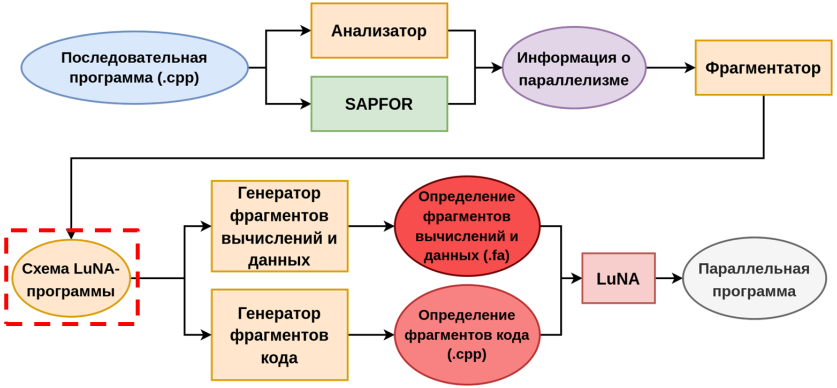


Схема LuNA-программы

Схема LuNA-программы содержит принципиальные решения о том, как LuNA-программа должна быть устроена, без технических деталей.

Схема LuNA-программы содержит информацию о базовых частях имен **фрагментов данных**.



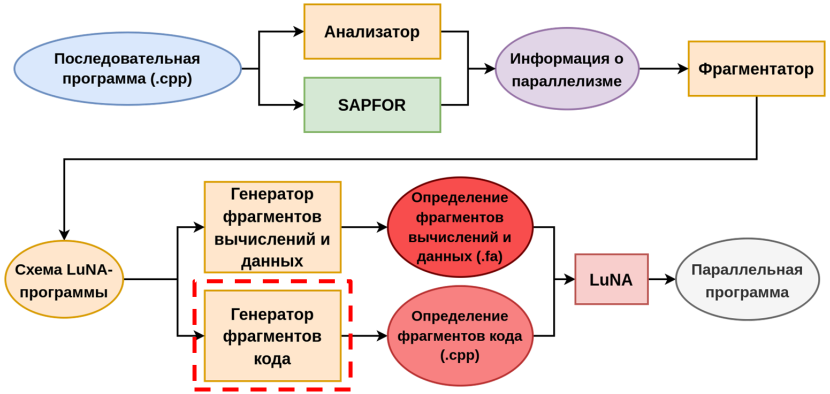
Для хранения схемы LuNA-программы разработан формат на основе JSON.

Генератор фрагментов кода

Задачей генератора фрагментов кода является **определение фрагментов кода**.

Генератор определяет фрагменты кода, описанные в схеме LuNA-программы. В эти фрагментах кода он добавляет операторы, описывающие:

- Инициализацию выходных фрагментов данных
- Получение значений входных фрагментов данных
- Уничтожающее потребление входных фрагментов данных выходными
- Доступы к элементам массивов
- Фрагментацию по пространству
- Инициализацию теневых граней



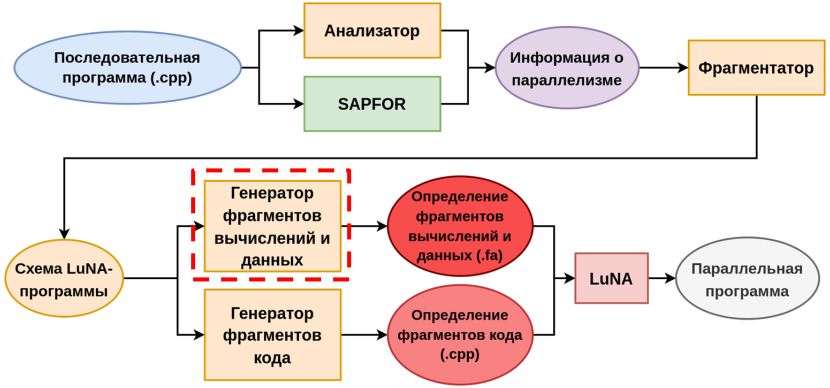
Генератор определяет фрагменты кода, выполняющие уничтожающее потребление, редукционные операции и объединение теневых граней.

Генератор определяет некоторые вспомогательные процедуры, в том числе вспомогательные для фрагментации по пространству.

Генератор фрагментов вычислений и данных

Задачей генератора фрагментов вычислений и данных является **определение фрагментов вычислений и данных**. Для этого генератор:

- Определяет операторы, описывающие применения фрагментов кода, представленных в схеме LuNA-программы.
- Добавляет во фрагменты вычислений аргументы, необходимые для выполнения фрагментации по пространству и инициализации теневых граней.
- Объявляет базовые имена фрагментов данных, которые описаны в схеме LuNA-программы.
- Определяет временные индексы фрагментов данных.
- Определяет операторы, описывающие фрагментацию по пространству.
- Определяет массовые операторы, соответствующие циклам исходной программы.
- Обеспечивает упорядоченное выполнение тех фрагментов вычислений, для которых это необходимо.
- Определяет операторы, описывающие объединение теневых граней.



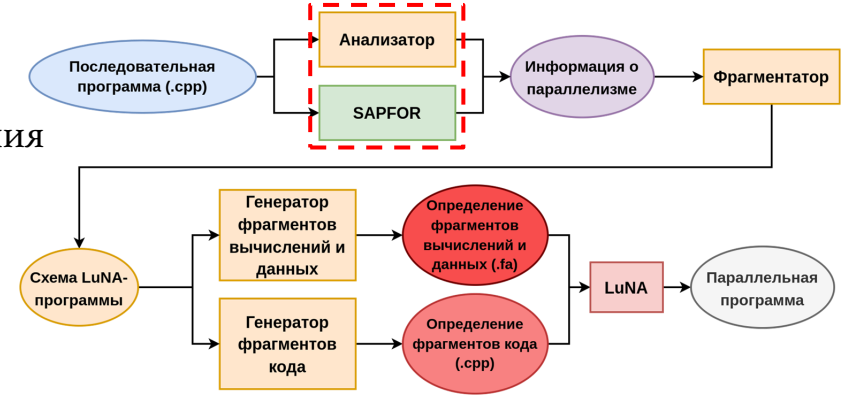
Выявление информации о параллелизме

SAPFOR ответственен за сбор информации о:

- Циклах
- Переменных
- Информационных зависимостях
- Редукционных операциях

Анализатор ответственен за сбор информации о:

- Выражениях, влияющих на граф потока управления
- Доступах к элементам массивов
- Выделениях памяти под массив
- Вызовах процедур
- Последовательных участках кода
- Использованиях индукционных переменных в телах циклов



Реализовывать алгоритмы анализатора предлагается с помощью библиотек LibTooling и LibASTMatchers, которые являются частью библиотеки компилятора Clang.

Реализация генератора фрагментов кода

Генератор фрагментов кода был реализован и протестирован на примере программы, реализующей метод Якоби.
Результаты работы исходной программы и сгенерированной программы оказались одинаковыми.

Время работы последовательной программы (с)	Время работы LuNA-программы (с)	Ускорение
70,42	10,76	**6,54**

Генератор сработал правильно, сгенерированная LuNA-программа существенно производительнее исходной последовательной.

Тестирование проводилось на ноутбуке ASUS Vivobook X1704ZA-AU123 с центральным процессором Intel Core i5-1235U.

Апробация

Работа была апробирована на следующих школах, семинарах и конференциях:

1. Всероссийская летняя XLI молодежная Школа-конференция по параллельному программированию с международным участием. Разработка и реализация алгоритмов автоматической генерации LuNA-программ из последовательных программ на основе дополнительной информации о возможности их распараллеливания и фрагментации (РИНЦ).
2. Зимняя школа по параллельному программированию 2024.
3. Семинар “Математическое обеспечение высокопроизводительных вычислительных систем” ИВМиМГ 2023.
4. Конференция “Молодежные Марчуковские научные чтения 2023” (ММНЧ 2023). Определение структуры представления информации о параллелизме в последовательной программе в модели вычислений потока управления для автоматического конструирования LuNA-программ с использованием системы SAPFOR (РИНЦ).
5. Всероссийская научная конференция с международным участием “Параллельные вычислительные технологии 2024” (ПаВТ’2024). В.К. Синюков, В.А. Перепелкин. Разработка и реализация алгоритмов автоматического конструирования LuNA-программ с использованием системы SAPFOR (с. 147–157) (РИНЦ).
6. Международная научная студенческая конференция (МНСК-2024). Разработка алгоритмов генерации фрагментов кода по их описанию в модели вычислений фрагментированного программирования для автоматического конструирования LuNA-программ (диплом 3-ей степени).

Заключение

Разработан подход к автоматическому конструированию LuNA-программ из последовательных программ с использованием системы SAPFOR.

На **защиту** выносятся следующие **положения**:

1. Подход к автоматическому конструированию LuNA-программ на основе системы SAPFOR.
2. Представления информации о программе, необходимой для ее распараллеливания и для генерации LuNA-программы. Форматы хранения этих представлений в компьютере.
3. Алгоритмы генерации LuNA-программ и алгоритмы определения информации, необходимой для этой генерации. Реализация алгоритмов генерации фрагментов кода.

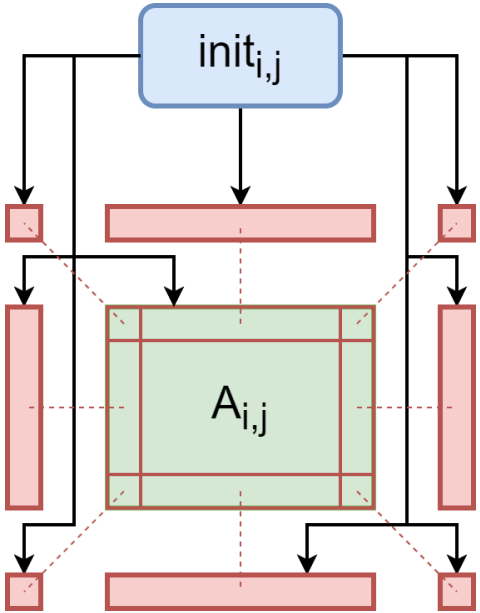
В дальнейшем планируется:

- Реализовать алгоритмы генерации фрагментов вычислений и данных, а также алгоритмы определения информации, необходимой для генерации LuNA-программ.
- Развить разработанные алгоритмы и представления таким образом, чтобы они подходили для более широкого класса задач.

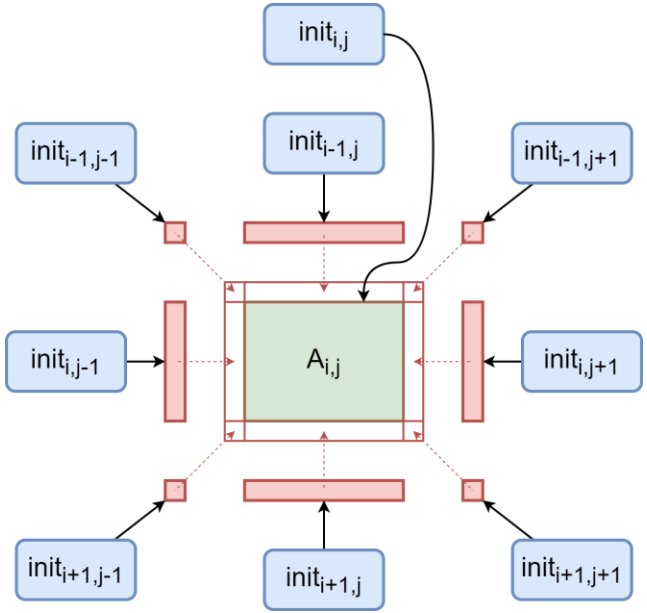
Теневые грани

Инициализация и объединение теневых граней массива A.

Инициализация



Объединение



Синие прямоугольники — фрагменты вычислений, зеленые — массив A, красные — теневые грани