Статический анализ кода пакетов прикладных программ для поиска протяженных участков для замены на параллельный эквивалент

Введение

Вопрос о повышении эффективности работы существующего кода можно решать несколькими способами, например:

- Распараллеливание кода.
- Адаптация кода под конкретную архитектуру путем замены некоторых фрагментов кода на другие более оптимальные реализации.
- Рефакторинг кода.

Терминология

- *Дубликаты* – два функционально эквивалентных фрагмента кода.
- Под *методом* будет подразумеваться один из алгоритмических способов решения конкретной задачи.
- Каждый метод может иметь несколько *мотивов* – конкретных вариантов реализации, которые разрабатываются в зависимости от возможностей вычислительной системы и алгоритма, используемого разработчиком.
- *Код пакета прикладных программ* (код ППП) – реализация пакета прикладных про- грамм, в которой необходимо обнаружить дубликат искомого мотива.
- *Шаблон* – искомый мотив, по которому непосредственно будет осуществляться поиск в коде ППП.

Описание предлагаемого подхода

- Создание базы фрагментов кода, на основании которой будет производиться поиск.
- Создание базы оптимальных функциональных эквивалентов для предложения замены найденных фрагментов.
- Осуществление поиска схожих фрагментов по созданной базе в каком-либо пакете прикладных программ.

Создание базы исходного кода для поиска

	OpenFOAM	Gromacs	MPI-ESM
Общее количество файлов	15,639	4,563	3,990
Общее количество строк	4,517,786	3,006,701	4,034,054
Количество строк кода в файлах типа .c, .f*	1,626,725	2,246,960	1,018,752

Создание базы исходного кода для поиска

В качестве искомых мотивов будут использованы реализации математических методов, однако сам алгоритм может быть использован и для других методов.
В качестве реализаций математических методов было принято решение использовать следующие библиотеки:

- ATLAS(Automatically Tuned Linear Algebra Software) – реализация BLAS для языков C и Fortran. Средний объем методов 268 строк.
- OpenCV (Open Source Computer Vision Library) – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения для языка C. Средний объем методов 297 строк.

Построение решения для поиска схожих фрагментов

- Создание списка зависимостей, требуемых мотивом.
- Обнаружение рекурсий.
- Генерация нового файла, включающего все необходимые вызовы процедур.
- Удаление незначащих фрагментов.

Предобработка исходного кода пакета прикладных программ

- Анализ пакета: обнаружение и составление списка файлов реализации на конкретном языке.
- Кластеризация файлов: отобранные файлы разбиваются на множество кластеров по наличию зависимостей для поэтапного сравнения с шаблоном.
- Удаление комментариев: до кластеризации или после, отдельно для каждого кластера.

Создание комплексной метрики для определения близости фрагментов

1. Фрагментом кода в заданном файле f будем считать последовательность лексем L_i, однозначно определяемую парой $\langle l_{begin}, l_{end} \rangle$, где l_{begin} и l_{end} – номера строк начала и конца фрагмента соответственно.
2. $\mathcal{P} = \{\mathcal{P}_i | 1 \leq i \leq N_p\}$ – множество исходного кода в пакете прикладной программы,
3. $M = \{M_j | 1 \leq j \leq N_m\}$ – множество реализаций из коллекции искомых фрагментов.
4. Тогда для двух фрагментов $L_{\mathcal{P}_i}$ и L_{M_j} зададим функцию $\rho(L_{\mathcal{P}_i}, L_{M_j})$ – расстояние между ними.
5. Фрагмент $L_{\mathcal{P}_i}$ будем называть дубликатом фрагмента L_{M_j} в отношении φ, если $\rho(L_{\mathcal{P}_i}, L_{M_j}) \leq \varphi \leq 1$.

Реализация алгоритма с использованием выбранных метрик

- На данном этапе был выбран способ вычисления метрики схожести на основании абстрактных синтаксических деревьев[1]:

$$S\left(L_{\mathcal{P}_i}, M_j\right) = \frac{2 \cdot N}{2 \cdot N + L + R}, \tag{1}$$

где N — число совпадающих узлов, L — число различных узлов в первом поддереве, R — число различных узлов во втором поддереве.

- Построение деревьев производилось с помощью генератора парсеров ANTLR.

[1] Baxter I. D. et al. Clone detection using abstract syntax trees

Результаты работы

- При кластеризации использовалась глубина, равная 3.
- Время указано в секундах

	OpenFOAM	Gromacs	MPI-ESM
Время анализа файлов	3.06	0.815	0.862
Количество обработанных файлов	8336	3206	1884
Время кластеризации	203.475	93.467	53.48
Количество кластеров	3643	2484	1167
Время удаления комментариев	257.103	121.467	85.225
Количество удаленных строк	15975	5775	5280

Результаты работы

- Описанный алгоритм поиска фрагментов был применен на пакете MPI-ESM. В качестве базы, на основании которой производился поиск, была создана коллекция реализаций численных методов, наиболее часто встречающихся в пакете
- Поиск осуществлялся при пороговом значении метрики 0.7

Количество найденных функций

	Метод Ньютона	Метод Гаусса	Метод релакса-ции	Метод простой итерации	Метод Ричардсона
Вызовы функций	5	11	2	6	5
Схожие реализации	12	28	15	8	3

Дальнейшая работа

- Создание базы шаблонов параллельных реализаций некоторых методов, которые будут предложены пользователю в качестве замены
- Реализация алгоритма выдачи информации о всех найденных схожих шаблонах и рекомендаций о замене и подстановке оптимальных эквивалентов
- Рассмотреть возможность интеграции такой системы в системы автоматического распараллеливания кода

Спасибо за внимание!