УДК 528.71
DOI 10.33764/2618-981X-2024-4-1-183-187

В. С. Симонов[1], М. С. Хайретдинов[2]*

Препроцессор языка программирования Python для решения задач мониторинга процессов в литосфере земли

[1] Новосибирский государственный технический университет, г. Новосибирск, Российская Федерация
[2] Институт вычислительной математики и математической геофизики СО РАН, г. Новосибирск, Российская Федерация
* e-mail: simonws@ya.ru

Аннотация. Эволюция программного обеспечения обуславливает переход от последовательных архитектур к распараллеленным и масштабируемым архитектурам, что объясняется необходимостью использования вычислительной мощности современных многоядерных и распределенных систем. Для автоматизации этого перехода предложен специальный препроцессор, который по определенным правилам преобразует последовательный код в форму, подходящую для платформ параллельной обработки данных. Основной целью создания препроцессора Python является преобразование фрагментов последовательного кода, таких как циклический переход по списку, в аналогичные фрагменты, которые можно обработать в современных вычислительных системах. Прозрачность и простота использования языка Python делают его удобной платформой для разработки такого инструмента. Препроцессор уже включает в себя функции Spark и использует сторонние библиотеки для упрощения и автоматизации преобразования кода. Это существенно облегчает задачи разработчиков, делая крупномасштабные вычисления доступнее для более широкого спектра приложений. В результате, данные методологии и инструменты помогают переходу к новым парадигмам разработки программного обеспечения, делая это более эффективным и доступным для разработчиков различных уровней. В данной статье представлен препроцессор Python, предназначенный для автоматизации преобразования из последовательных приложений на Python в такие, которые используют возможности крупномасштабных вычислительных сред.

Ключевые слова: формальные языки, распределенные вычисления, мониторинг процессов

V. S. Simonov[1], M. S. Khairetdinov[2]*

Python programming language preprocessor for solving problems of monitoring processes in the Earth's lithosphere

[1] Novosibirsk State Technical University, Novosibirsk, Russian Federation
[2] Institute of Computational Mathematics and Mathematical Geophysics of Siberian Branch of Russian Academy of Sciences, Russian Federation
* e-mail: simonws@ya.ru

Abstract. The evolution of software causes the transition from sequential architectures to parallelized and scalable architectures, which is explained by the need to use the computing power of modern multicore and distributed systems. To automate this transition, a special preprocessor has been proposed, which, according to certain rules, converts serial code into a form suitable for parallel data processing platforms. The main purpose of creating a Python preprocessor is to convert fragments of sequential code, such as looping through a list, into similar fragments that can be processed in modern

computing systems. The transparency and ease of use of Python make it a convenient platform for developing such a tool. The preprocessor already includes Spark functions and uses third-party libraries to simplify and automate code conversion. This greatly facilitates the tasks of developers, making large-scale computing more accessible to a wider range of applications. As a result, these methodologies and tools help the transition to new software development paradigms, making it more efficient and accessible to developers at various levels. This article presents a Python preprocessor designed to automate the conversion from sequential Python applications to those that use the capabilities of large-scale computing environments.

Keywords: formal languages, distributed computing, process monitoring

Введение

В области разработки программного обеспечения переход от последовательных или однопоточных приложений к приложениям, использующим крупномасштабные вычислительные ресурсы, таким как многопоточные, распределенные или облачные среды, сопряжен с рядом трудностей. Тем не менее, это необходимая эволюция, позволяющая справляться с растущими объемами данных и вычислительными требованиями современных приложений. Последовательные приложения работают поэтапно, выполняя одну операцию за раз. Этот подход, несмотря на свою простоту, становится все более неадекватным для решения задач, требующих обработки больших наборов данных или выполнение сложных вычислений в разумные сроки [1].

MapReduce - популярная модель для разработки эффективных приложений. Она отличается разнообразием и высокой эффективностью работы с такими реализациями [2,3]. Переход к парадигмам крупномасштабных вычислений, таким как параллельная обработка, распределенные вычисления и облачные вычисления — может значительно сократить время выполнения за счет разделения задач на более мелкие блоки, которые могут выполняться одновременно.

Методы и материалы

Препроцессор Python, разработанный в данной работе, представляет собой инструмент, который анализирует и преобразует код Python, написанный для последовательного выполнения, в форму, которая может быть выполнена в параллельной или распределенной среде. Он включает в себя функции анализа кода, распознавания образов и автоматического преобразования кода.

Препроцессор, предназначенный для преобразования входного кода в эффективный код в рамках парадигмы MapReduce, поддерживаемый платформой PySpark. Конвейер выполнения состоит из трех основных модулей:

Анализатор кода. Этот начальный этап включает в себя разбор входного кода с помощью абстрактного синтаксического дерева (AST) и выполнение статического анализа программы. Цель этого этапа - определить сегменты кода, которые подходят для поиска эквивалента. Целью являются циклы, содержащие итерации по структурам данных. Во время прохождения AST выбираются условные выражения, функции, пользовательские типы и циклы. Для каждого идентифицированного цикла анализатор выполняет две задачи: построения промежу-

точного представления программы (IR) для последующего синтеза; определения условий верификации для проверки эквивалентности шаблона кода исходному фрагменту кода.

Генератор сводок. Сводка программы - это постусловие входного фрагмента кода, описывающее состояние программы после выполнения [4]. На данном этапе препроцессор синтезирует резюме программы на основе пространства поиска и VC, предоставляемых анализатором программ [5]. Модуль реализует основную логику для преобразования высокоуровневой программной логики в форму, которая легко преобразуется в исполняемый код для платформ параллельной обработки данных.

Генератор кода. Как только будет найдена и проверена сводка программы, последним шагом будет преобразование IR в исполняемый код. Этот модуль легко адаптируется и поддерживает генерацию кода для платформы map-reduce: Spark. Данная функция позволяет осуществлять динамический выбор между различными стратегиями реализации на основе показателей производительности в реальном времени, что дополнительно оптимизация эффективности выполнения переведенных программ [6].

Реализация включает в себя разработку механизма синтаксического анализа кода Python, набора правил преобразования для преобразования последовательных шаблонов в параллельные конструкции. Инструмент реализован с использованием абстрактных синтаксических деревьев (AST) для анализа и преобразования кода.

Мы внедрили препроцессор, используя инфраструктуру компилятора ROSE [7], для преобразования кода Python в абстрактное синтаксическое дерево (AST). Он может эффективно анализировать код Python в AST. AST — это древовидное представление структуры исходного кода, которое абстрагируется от синтаксических деталей, фокусируясь на иерархической и логической структуре синтаксиса. Препроцессор может обрабатывать все основные операции скрипта Python, включая арифметические, логические и разрядные операции. Он также поддерживает манипуляции с примитивными массивами и общими интерфейсами сбора данных. Возможно расширение препроцессора для работы с другими структурами данных, такими как стек или очередь.

Результаты

Результаты работы использовались для построения корреляционных сверток сейсмотрасс. Образцы исходных записей представлены на рисунке 1. Полученные записи сворачиваются с опорным сигналом в соответствии с алгоритмом [8] с применением процедуры БПФ. Опорный сигнал синтезируется в точке приема.

Полученные после корреляционной свертки сейсмотрассы представлены на рисунке 2.

Результаты работы последовательной реализации алгоритма совпадают с результатами работы распределенного приложения.

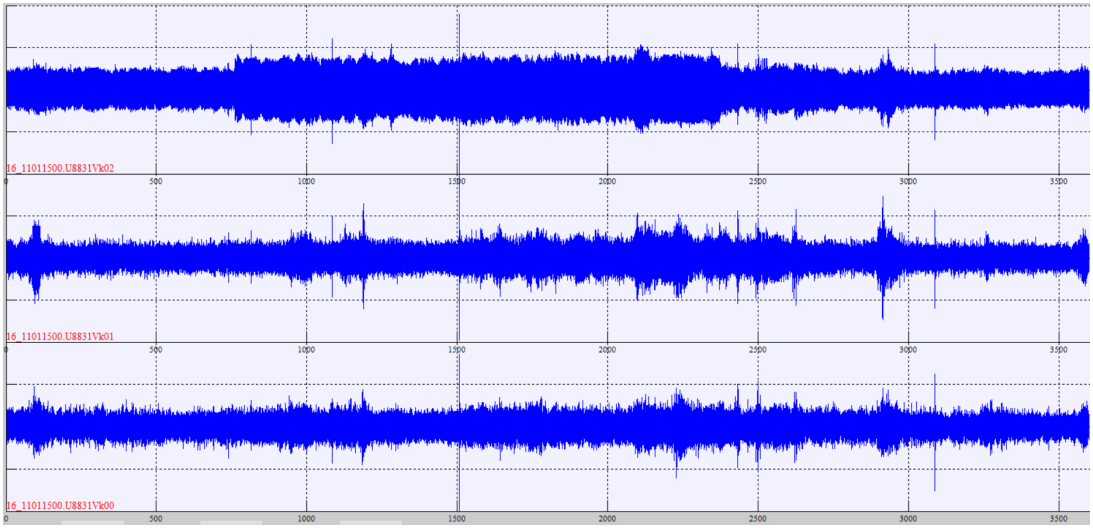

Рис. 1. Исходные трассы

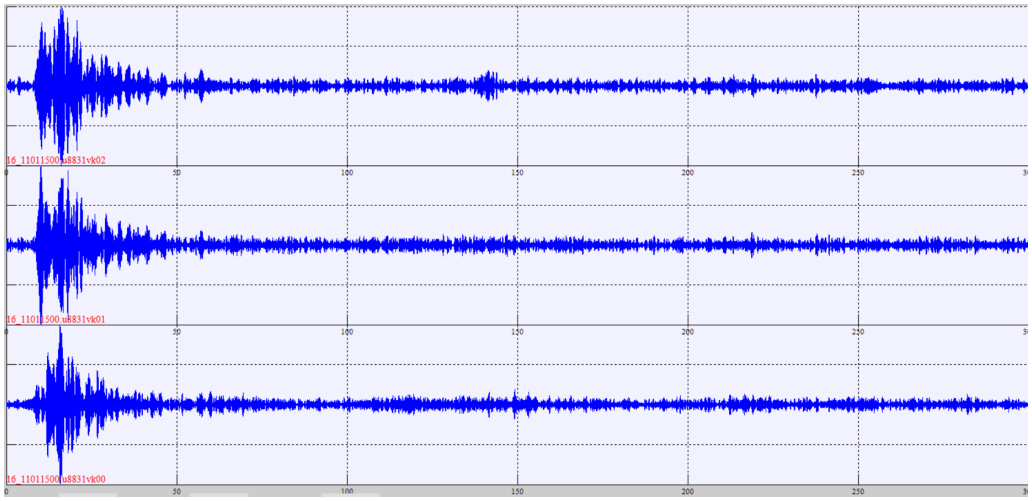

Рис. 2. Сейсмотрассы после корреляционной свертки

Обработка алгоритма [8] препроцессором не стала проблемой, но интересно рассмотреть разработанное решение с другой стороны. В таблице 1 показано среднее время (более 5 запусков), необходимое для обобщения сводных данных по каждому из четырех других алгоритмов.

Таблица 1

Результаты экспериментов

Показатель	Анализ программы	Синтез	Итерации
Сложение	~1 с	13 с	1
Подсчет слов	~1 с	45 с	1
Вхождение строки	~1 с	1195 с	2
3D-гистограмма	~1 с	1976 с	2

Компилятор синтезировал реализации Spark для некоторых тестов в течение часа. Более простые тесты, такие как суммирование и подсчет слов, были преобразованы менее чем за минуту и потребовали всего одной итерации генерации грамматики. Ни один тест не требовал более двух итераций для успешного выполнения, что показано в таблице 1.

Заключение

Препроцессор, разработанный в рамках этого исследования, представляет собой значительный шаг на пути к тому, чтобы сделать крупномасштабные вычисления более доступными для широкого спектра приложений. Результаты показывают, что настоящий инструмент подходит и для решения задач мониторинга процессов в литосфере земли. Автоматизируя процесс преобразования, данный инструмент не только упрощает переход от последовательной архитектуры к параллельной и распределенной, но и открывает новые возможности для повышения производительности и масштабируемости. По мере дальнейшего развития крупномасштабных вычислений такие инструменты, как этот препроцессор, будут играть решающую роль, позволяя приложениям использовать весь потенциал современных вычислительных инфраструктур.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. J. Dean, S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters: Commun. ACM 51, 1. - 2008, - 107–113 с.

2. Apache Spark. (2018). URL: https://spark.apache.org (дата обращения: 14.05.2024)

3. A.V. Aho, M.S. Lam, R. Sethi, J. D. Ullman. Compilers: Principles, Techniques, and Tools (2Nd Edition). Addison-Wesley Longman Publishing Co., Inc.. - 2006. - 38–39 с.

4. C. A. R. Hoare. An Axiomatic Basis for Computer Programming. Commun. ACM 12, - 10 Oct. 1969. - 577–579 с.

5. R. Bodik, B. Jobstmann. 2013. Algorithmic program synthesis: introduction. International Journal on Software Tools for Technology Transfer. -2013, - 399–403 с.

6. V. S. Simonov, M. S. Khairetdinov. Automatic decomposition of a sequential algorithm for MapReduce Frameworks // International multi-conference on engineering, computer and information sciences (SIBIRCON–2022): proc., Novosibirsk–Yekaterinburg, 11–13 Nov. 2022, IEEE. - 2022. - 1780–1783 с.

7. ROSE. An open source compiler infrastructure. URL: http://rosecompiler.org/ROSE HTML Reference/index.html (дата обращения: 18.03.2024).

8. Хайретдинов М. С., Воскобойникова Г. М., Седухина Г. Ф. Алгоритмы поточной свертки в задачах активного вибросейсмоакустического мониторинга // Интерэкспо Гео-Сибирь. 2017. №1. - 2017.