Марков Никита Олегович,
Студент 4 курса
ФГАОУ ВО «МГТУ имени Н.Э.Баумана»
г. Москва

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ЭФФЕКТИВНОСТИ МЕТОДОВ УСКОРЕНИЯ ВЕБ-СКРАПИНГА В PYTHON

Аннотация. В статье рассматриваются методы ускорения веб-скрапинга на языке Python с использованием асинхронного программирования (модуль asyncio) и многопоточности (модуль concurrent.futures) на примере ресурса books.toscrape.com. Проведён сравнительный эксперимент, включающий тестирование последовательной, пакетной, параллельной и комбинированных стратегий отправки запросов, а также их модифицированных вариантов с сохранением сессии и динамическим ожиданием загрузки элементов страницы. Оценка производительности велась по времени отправки запроса, масштабируемости и потреблению оперативной памяти (ОП). Результаты показали, что наибольшую скорость при масштабировании обеспечивает параллельная асинхронная отправка запросов в нескольких веб-драйверах, однако она требует значительных ресурсов ОП и сложна в реализации. Последовательная отправка запросов наиболее экономична по памяти, но уступает в скорости. Сделан вывод о зависимости выбора подхода от целей и ограничений проекта.

Ключевые слова: веб-скрапинг, Python, асинхронное программирование, многопоточность, Selenium, веб-драйвер, масштабируемость, оперативная память.

В современном цифровом мире для исследования рынка, мониторинга цен на продукты и услуги из-за быстро меняющихся данных всё больше организаций вынуждены использовать средства автоматизированного сбора данных с веб-сайтов (веб-скраперы). Однако при необходимости постоянной обработки больших объёмов информации появляется проблема недостаточной скорости

работы скраперов, которые не способны обработать растущие объёмы данных. Из-за этого возникает необходимость в ускорении средств для сбора данных с веб-сайтов, чтобы обеспечить своевременное получение актуальной информации и повысить эффективность аналитических процессов.

В статье рассматриваются подходы, основывающиеся на двух основных парадигмах конкурентного программирования в языке Python. С одной стороны, асинхронное программирование, реализуемое через модуль asyncio, позволяет отправлять большое количество запросов на сервер, не дожидаясь получения каждого ответа и не блокируя основной поток выполнения программы, что должно привести к существенному ускорению сбора данных. С другой стороны, многопоточность, основанная на модуле concurrent.futures, может обеспечить параллельную отправку большого числа запросов, что также должно значительно увеличить эффективность скрапинга. Для извлечения данных из веб-страниц использованы библиотеки Selenium и BeautifulSoup4, что соответствует практикам, описанным в источнике [1].

Целью исследования является сравнительная оценка способов автоматизированного сбора данных с веб-сайтов на примере веб-сайта books.toscrape.com для облегчения планирования разработки веб-скраперов под различные сценарии использования. Задачи исследования:

- Рассмотреть различные подходы к ускорению веб-скрапинга в Python.
- Разработать экспериментальную методику для сравнения эффективности выделенных подходов.
- Провести эксперимент, включающий замеры метрик скорости отправки запросов, качества масштабирования и объема потребляемой ОП для каждого способа на разном количестве отправляемых запросов.
- Провести сравнительный анализ полученных результатов и определить наиболее эффективные подходы и их комбинации.

Актуальность исследования обусловлена необходимостью оптимизации веб-скраперов, взаимодействующих с большими объемами данных, которые постоянно должны поддерживаться в актуальном состоянии.

Для проведения эксперимента была разработана программа на языке Python, в которой описаны функции, реализующие все тестируемые подходы к ускорению скрапинга [1].

Базовой функцией, оптимизация которой проводилась в рамках исследования, является функция, реализующая последовательную (синхронную) отправку веб-запросов. Алгоритм работы этой функции описывается следующим образом: для каждой URL из массива, переданного в качестве параметра функции, веб-драйвер Selenium устанавливает соединение с веб-сервером, переходит по ссылке, ожидает загрузки страницы в течение фиксированного временного интервала (для тестирования было выбрано значение в 3 секунды как компромисс между скоростью и точностью скрапинга), получает исходный код страницы, после чего сохраняет содержание одного из блоков обработанной веб-страницы, полученное при помощи библиотеки BeautifulSoup4.

Асинхронная отправка запросов реализована тремя функциями, описывающими принципиально разные подходы. Одна из них реализует последовательную отправку, при которой после формирования первого запроса он сразу отправляется, после чего начинается формирование следующего запроса, что должно привести к замедлению работы скрапера из-за необходимости в обработке прерываний, вызванных получением ответа от сервера, во время формирования очередного запроса. Другая функция реализует пакетную отправку, то есть запрос не отправляется сразу после формирования, а ожидает окончания формирования последнего запроса в пакете, после чего запросы отправляются последовательно. Последняя функция реализует параллельную асинхронную отправку запросов в нескольких веб-драйверах [2].

Многопоточная отправка запросов заключается в параллельном выполнении базовой функции для нескольких ссылок. Максимальное количество одновременно обрабатываемых ссылок определяется в модуле concurrent.futures и может быть задано вручную, однако предварительное тестирование показало, что автоматическое определение этого значения, реализованное в рамках модуля,

является наиболее оптимальным по соотношению скорости и использования ресурсов операционной системы.

Также протестированы способы, включающие отправку всех запросов в рамках одной сессии [3], без разрыва соединения с сервером, а также замена фиксированного временного интервала ожидания загрузки страницы на динамический интервал ожидания конкретного элемента веб-страницы, реализованная при помощи библиотеки Selenium [4].

Экспериментальная установка включала следующие компоненты:

Аппаратное обеспечение: шестиядерный процессор Ryzen 5 5500U, 16 ГБ оперативной памяти.

Программное обеспечение: Python 3.10, BeautifulSoup4 4.10, Selenium 4.28, операционная система Linux Mint 21.2 Cinnamon.

Тестовые данные: каталог книг, размещённый на веб-сайте books.toscrape.com.

Первый тестовый сценарий, предназначенный для выбора нескольких наиболее эффективных способов ускорения скрапинга, представлял собой обработку 60 веб-страниц, соответствующих определенным позициям каталога книг, описанными функциями, причём каждая функция обрабатывала ссылки в цикле из 5 повторений.

Второй тестовый сценарий отличен от первого тем, что тестирование проводилось только для отобранных эффективных способов. Количество обрабатываемых страниц последовательно увеличивалось для проверки масштабируемости алгоритмов, соответствующих выбранным способам. Для сравнения эффективности способов использовались такие метрики, как количество запросов, посылаемых алгоритмом за единицу времени, а также объём оперативной памяти, необходимый для нормальной работы алгоритма.

Результаты экспериментов представлены на рисунке 1, где показана диаграмма зависимости общего времени отправки 60 запросов от каждого способа скрапинга. Для пакетной и параллельной асинхронной отправки запросов указано лучшее полученное время. В таблице 1, приведённой ниже, для

каждого способа рассчитан коэффициент T — коэффициент ускорения скрапинга относительно базового способа, и среднее время одного запроса.

На примере метода последовательной отправки запросов можно заметить, что использование одной HTTP-сессии [3] позволяет сократить время выполнения скрапинга почти вдвое, а замена явного ожидания на неявное — ещё в несколько раз [4]. Стоит отметить, что метод последовательной отправки с описанными усовершенствованиями на тестовых данных показал почти лучший результат, проиграв в скорости только методу параллельной асинхронной отправки запросов.

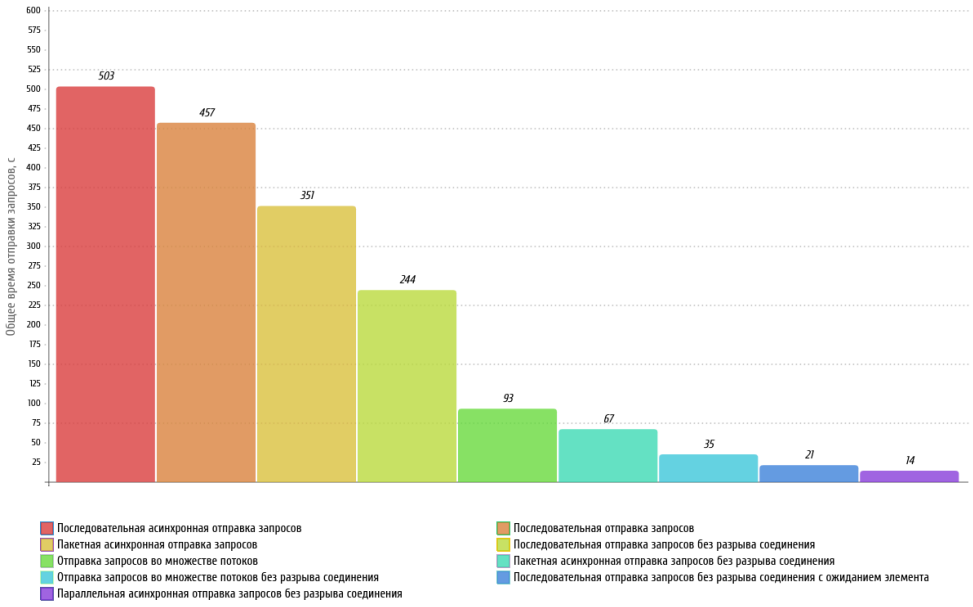

Рисунок 1 — Диаграмма зависимости времени отправки 60 запросов от способа скрапинга

Таблица 1

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ВРЕМЕНИ ОТПРАВКИ ЗАПРОСОВ РАЗЛИЧНЫМИ СПОСОБАМИ

Способ отправки запросов	**$t_{ср}$, с**	**T**	**$t_{запр}$, с**
Последовательная асинхронная	503,09	0,91	8,38
Последовательная	457,24	1	7,62
Пакетная асинхронная	351,85	1,3	5,86
Последовательная без разрыва соединения	244,18	1,87	4,07
Во множестве потоков	93,73	4,88	1,56

Пакетная асинхронная без разрыва соединения	67,13	6,81	1,12
Во множестве потоков без разрыва соединения	35,92	12,73	0,6
Последовательная без разрыва соединения с ожиданием элемента	21,77	21	0,36
Параллельная асинхронная без разрыва соединения	14,38	31,8	0,24

Анализируя представленные результаты, необходимо понимать, что оптимизация веб-скраперов необходима тогда, когда количество отправляемых им запросов на один или несколько порядков превышают тестовое значение. Для анализа качества масштабирования наиболее эффективных способов приведены рисунки 2 и 3.

На рисунке 2 изображена диаграмма, отражающая зависимость скорости скрапинга (количества запросов, отправляемых в единицу времени) от общего количества отправляемых запросов. Можно заметить интересную особенность: если методы последовательной, многопоточной и пакетной асинхронной отправки запросов имеют почти неизменную скорость работы, то есть при масштабировании их эффективность остаётся неизменной, то метод параллельной асинхронной отправки запросов значительно увеличивает свою эффективность на большем количестве запросов. Этот эффект вызван тем, что в начале работы алгоритм некоторое время инициализирует веб-драйверы, во время чего не происходит отправки запросов, но сама отправка запросов происходит быстрее, чем в других рассмотренных способах. Можно также отметить, что от количества инициализируемых веб-драйверов (которое указано в скобках на диаграмме) зависит степень выраженности этого эффекта. Так, например, эффективность способа при инициализации 3 драйверов выше, чем при инициализации 5 драйверов, на малом количестве запросов, что объясняется меньшим временным интервалом, затрачиваемым на инициализацию, однако на большем количестве запросов эффективнее оказывается другой способ, поскольку параллельная отправка в 5 веб-драйверах происходит быстрее.

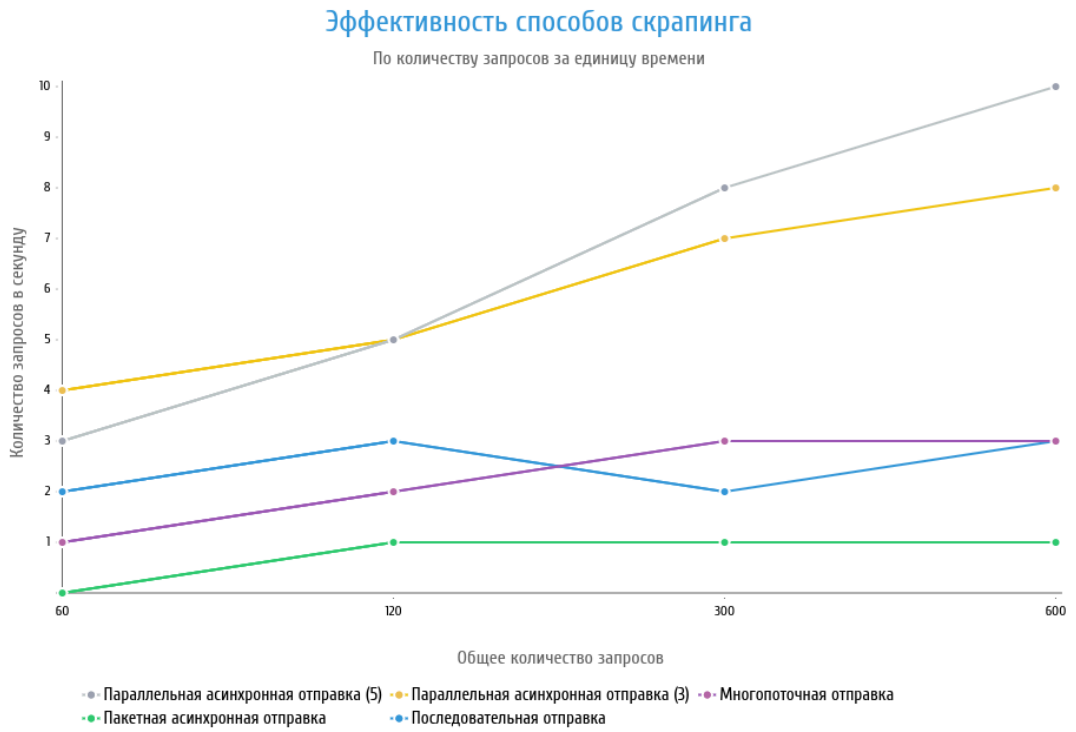

Рисунок 2 — Диаграмма эффективности скрапинга по скорости отправки запросов

Таким образом, наибольшую эффективность по скорости отправки запросов при масштабировании имеет способ параллельной асинхронной отправки запросов. В тестовом сценарии на указанном аппаратном обеспечении наибольшую эффективность показал способ, реализующий отправку запросов в 5 веб-драйверах, однако это число может зависеть как от аппаратного, так и от программного обеспечения и отличаться в других сценариях.

На рисунке 3 изображена диаграмма эффективности выбранных способов скрапинга по количеству используемой оперативной памяти. Можно увидеть, что более медленные способы занимают незначительный объём оперативной памяти, тогда как способ параллельной асинхронной отправки запросов в несколько раз более требователен к ОП, причём требования растут при увеличении количества используемых одновременно веб-драйверов. Более того, поскольку при увеличении числа используемых веб-драйверов увеличивается и нагрузка на операционную систему (ОС), которая поддерживает их работу, при достаточно большом количестве драйверов ОС из-за постоянного переключения между ними может значительно снизить скорость отправки запросов, из-за чего особенно

важно подбирать оптимальное число используемых одновременно драйверов. Таким образом, наиболее эффективным способом по использованию ОП является простая последовательная отправка запросов.

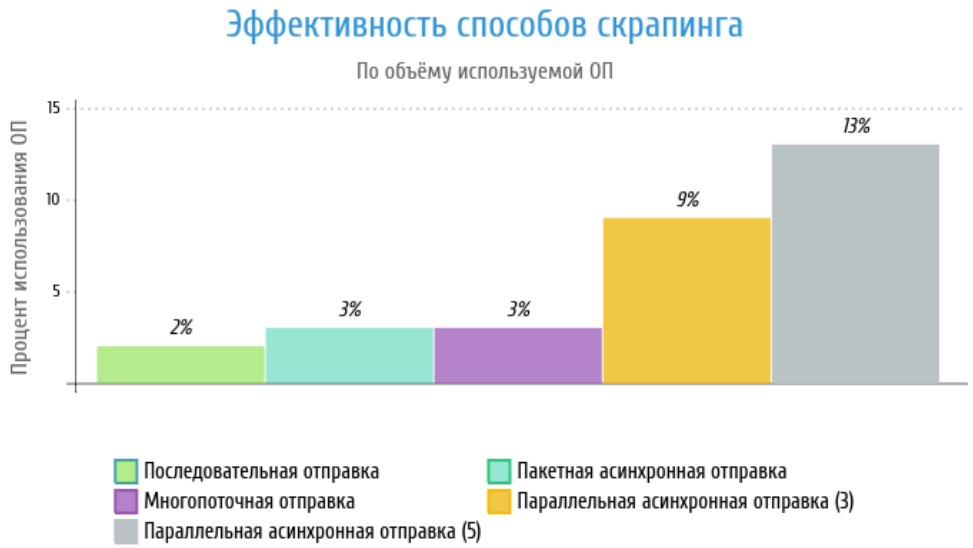

Рисунок 3 — Диаграмма эффективности скрапинга по использованию ОП

Стоит также отметить сложность программной реализации рассмотренных способов. Метод последовательной отправки запросов является самым простым ввиду отсутствия необходимости написания функций, способных выполняться независимо друг от друга. Методы многопоточной отправки и пакетной асинхронной отправки более сложны в реализации, однако самым трудоёмким является метод параллельной асинхронной отправки запросов, поскольку он требует полный переход к парадигме асинхронного программирования.

Подводя итоги проведённого исследования, можно сделать вывод, что каждый из рассмотренных способов эффективен по одному из выделенных критериев, однако нет такого, который эффективен по всем критериям сразу. Поэтому выбор метода веб-скрапинга напрямую зависит от задач, для выполнения которых разрабатывается программа. Если, например, веб-скрапер необходим как вспомогательный инструмент для анализа данных, который не сможет занимать значительный объём ресурсов ОС, однако должен обеспечить приемлемую скорость работы, то оптимальным методом стала бы последовательная отправка запросов. Если же скрапер является ядром системы, которое постоянно должно поддерживать значительные объёмы данных в

актуальном состоянии, то логичным выбором стал бы метод параллельной асинхронной отправки запросов.

В дальнейшем рекомендуется провести дополнительные исследования по оптимизации метода параллельной асинхронной отправки запросов, а также изучить изменение эффективности скрапинга при использовании других веб-драйверов, в особенности, позволяющих полностью следовать парадигме асинхронного программирования. Выявленные в данной статье результаты могут служить основой для разработки практических рекомендаций при выборе архитектурных решений для систем, работающих с большими объёмами данных, получаемых с веб-сайтов.

Список литературы:

1. Web Scraping with Python: сбор данных с современных веб-сайтов / P. Митчелл. — 2-е изд. — Пекин; Бостон: O’Reilly Media, 2018. — 379 с.

2. Организация параллельных асинхронных вычислений [Электронный ресурс]. – Режим доступа: https://docs.python.org/3/library/asyncio-eventloop.html (дата обращения: 01.08.2025)

3. Установление и управление HTTP-сессиями [Электронный ресурс]. – Режим доступа: https://w3c.github.io/webdriver/#new-session (дата обращения: 11.07.2025)

4. Явное и неявное ожидание в Selenium [Электронный ресурс]. – Режим доступа: https://www.selenium.dev/documentation/webdriver/waits/ (дата обращения: 23.07.2025)