УДК 004.41

ТЕСТИРОВАНИЕ ИНТЕРПРЕТАТОРОВ CPYTHON И PYPY ДЛЯ СОЗДАНИЯ ВЕБ-ПРИЛОЖЕНИЙ

Олимпиев Н.В. (Университет ИТМО)
Научный руководитель – к.пед.н., доцент Государев И.Б.
(Университет ИТМО)

Работа посвящена тестированию Python-интерпретаторов CPython и PyPy для создания веб-приложений. Предложенная в работе методика учитывает популярные веб-фреймворки: Flask, Django, Bottle, CherryPy, Falcon, а также следующие численные критерии сравнения интерпретаторов: тип запроса, количество запросов в секунду, количество подключений и время ожидания. В результате визуализированы полученные численные результаты и сделан вывод, что использование интерпретатора PyPy в большинстве случаев повышает производительность приложения.

Введение. Решить задачу увеличения производительности веб-приложения с наименьшими затратами ресурсов возможно с помощью смены интерпретатора языка. Сложность в том, что из-за растущего количества Python-имплементаций и веб-фреймворков для разработчиков все более острой проблемой становится выбор подходящего варианта на основе различных критериев. Во-первых, необходимо выбирать наиболее эффективный интерпретатор для конкретных задач. Во-вторых, смена интерпретатора может как кардинально улучшить, так и ухудшить производительность приложения.

Поскольку все сравнительные анализы интерпретаторов имеют короткий период актуальности из-за интенсивного процесса обновления версий, важно в первую очередь провести анализ существующих исследований, в частности проанализировать срок давности работ и полноту проведенных тестов. В рассматриваемых работах методика зачастую затрагивает общие сценарии использования приложения, многопоточность, ввод-вывод и метапрограммирование. Основные методы, используемые в научных трудах, это метод бенчмаркинга, профайлинга для анализа кода и выделения малопроизводительных участков кода. По результатам анализа источников можем сделать вывод, что труды либо носят общий характер исследования, либо специфический, при этом большинство авторов стараются охватить как можно больше имплементаций для тестирования, а версии рассматриваемых интерпретаторов зачастую не являются актуальными.

Решением проблемы выбора интерпретатора для веб-приложения является проведение независимого и воспроизводимого исследования с методологическим подходом, основанном на использовании Python-фреймворков. Для этого необходимо разработать методику таким образом, чтобы ее результаты могли быть практически применимы веб-разработчиками при выборе интерпретаторов Python.

Основная часть. Для формирования методики тестирования интерпретаторов выделим лучшие практики из существующих работ. Для сравнения возьмем результаты CPython как де-факто основного интерпретатора языка и JIT-реализацию PyPy. Основной упор в рамках исследования сделаем на количестве критериев сравнения, а не на количестве тестируемых интерпретаторов, поскольку нами поставлена цель разработать независимую и масштабируемую методику. Акцент нашего исследования сделаем на использовании Python в веб-разработке, для этого рассмотрим применение веб-фреймворков в качестве основного способа реализации приложений. Также при проведении тестирования не будем рассматривать Python 2, поскольку он достиг статуса «End Of Life». Для исследования возьмем актуальные версии 5 популярных Python-фреймворков, использующих WSGI: Django 3.2.2, Flask 1.1.2, Bottle 0.12.19, CherryPy 18.6.0 и Falcon 3.0.1.

Для проведения тестирования воспользуемся подходом к бенчмаркингу из репозитория «web-frameworks» от The Benchmarker, содержащего шаблоны приложений для фреймворков и описание предлагаемого авторами процесса тестирования. Для тестирования возьмем за основу шаблоны приложений из репозитория. Для каждого приложения полностью отключим

логирование для уменьшения дополнительной нагрузки на тестируемую систему. Тестирование будем проводить с помощью методов GET и POST с имитацией 64 и 512 подключений для каждого фреймворка. Для запуска веб-приложений будем использовать WSGI HTTP сервер Gunicorn с Meinheld в качестве типа воркера. Тип воркера Meinheld был выбран из-за легковесности реализации WSGI веб-сервера с поддержкой асинхронного ввода-вывода. Тестирование проведем с помощью построения окружения на базе Docker. Для замеров производительности интерпретаторов воспользуемся wrk – HTTP-инструментом для бенчмаркинга. Данный инструмент способен генерировать нагрузку при запуске на одном многоядерном ЦП. wrk взаимодействует со скриптами на языке Lua, благодаря чему мы имеем возможность рассчитать множество разнообразных метрик. Для достижения большей универсальности результатов тестирования определим следующий набор сравнительных критериев: количество запросов в секунду; время ожидания и его перцентили от 50% до 99.999%. При формировании метрик нами был учтен тот факт, что основным фактором при выборе имплементации динамического языка Python является производительность во время выполнения. Для каждого фреймворка и интерпретатора проведем по 4 замера с 1 потоком длительностью 60 секунд, каждый замер повторим 10 раз и посчитаем среднее арифметическое результатов.

Выводы. По результатам можем сделать вывод, что при отсутствии у веб-приложения особенностей, связанных с ограничениями PyPy (ограниченная поддержка расширений, написанных на Си; неэффективность с короткими программами и скриптами; неполная компилируемость PyPy), наилучшим вариантом является использование PyPy как основного интерпретатора для увеличения производительности приложения. При этом присутствует несколько исключений, связанных с фреймворками Flask, CherryPy и Falcon. Для данных случаев общей закономерностью является большое значение максимального времени ожидания и его перцентиль 90 и более процентов.

Полученные численные результаты могут быть использованы разработчиками при выборе подходящего способа интерпретации Python для веб-приложения на основе описанных нами критериев. Помимо этого, практическая, подтвержденная в рамках работы значимость данного исследования заключается в описании методики тестирования интерпретаторов. Несмотря на то, что проведенное исследование было сфокусировано на сравнении CPython и PyPy, разработанная методика является гибкой и масштабируемой за счет использования lua-скриптов и бенчмарк-инструмента wrk. Изменение параметров запуска выбранных нами инструментов позволяет получить любые дополнительные метрики для сравнения интерпретаторов. Методы в работе сформированы с учетом возможности их применения для сравнения любых требуемых веб-фреймворков и интерпретаторов Python. Как следствие, разработанная методика может быть использована для проведения расширенного тестирования интерпретаторов или комплексного сравнительного анализа имплементаций языка и Python-фреймворков.

Олимпиев Н.В. (автор) Подпись

Государев И.Б. (научный руководитель) Подпись