УДК 004.05

ИССЛЕДОВАНИЕ ВЛИЯНИЯ ИСПОЛЬЗОВАНИЯ СТАТИЧЕСКОЙ ТИПИЗАЦИИ И ТРАНСПИЛЯЦИИ КОДА НА ПРОИЗВОДИТЕЛЬНОСТЬ NODE.JS ФРЕЙМВОРКОВ

Демашов Д.С. (Национальный исследовательский университет ИТМО)**, Лактюшкин О. Г.**
(Национальный исследовательский университет ИТМО)
Научный руководитель – к.п.н., доцент Государев И.Б.
(Национальный исследовательский университет ИТМО)

Возможности экосистем Javascript и Node.js позволяют использовать различные вариации языка для ускорения разработки, однако такой способ организации работы приложений в большей степени внедряется при построении окружения клиентских систем и мало изучен при создании серверных приложений. Данная статья направлена на определение уровней производительности такого архитектурного подхода, учитывая особенности платформы Node.js с помощью проведения нагрузочного тестирования шаблонных приложения, реализованных на разных технологиях.

Введение. На текущий момент развития языка Javasript и платформы Node.js основными факторами производительности приложений принято считать набор модулей, на которых строится приложение, в том числе выбранные фреймворки, базы данных и способы построения клиентских интерфейсов. При этом в меньшей степени исследуется воздействие вариаций языка как на скорость разработки, так и на производительность серверных приложений.

Цель данной работы – идентифицировать факторы, влияющие на производительность серверных веб-приложений, реализованных на платформе JavaScript, и выявить тренды её изменения в условиях внедрения новых языковых функций в язык на разных уровнях.

Современный JavaScript подвергается изменениям на различных уровнях: общем (стандартном), экспериментальном и частном:

1. Ежегодные модификации стандарта, получившие название ES.Next.
2. Экспериментальные изменения, реализуемые производителями платформы.
3. Локальные изменения, вносимые в код разработчиками с использованием плагинов транспилятора, в том числе собственного изготовления.

С другой стороны, изменения вносятся в собственно процесс разработки, что может быть достигнуто на одном из языковых уровней и (или) на уровне организации процесса, например, при выборе инструментов. Примером такого влияния организации процесса разработки является использование инструментов линтинга и сборки.

Основная часть. Ставя своей целью добиться прироста производительности, можно выделить следующие основные факторы или аспекты:

1. Рост скорости исполнения кода.
2. Рост скорости разработки проекта.

Одна из основных сложностей в веб-разработке состоит в том, чтобы достичь баланса этих двух аспектов и затем отрегулировать их с тем, чтобы производительность увеличилась с учётом вносимых в язык и процесс разработки изменений. Так, при переходе к использованию нового инструментария наблюдается временный отрицательный прирост скорости разработки проекта за счёт преодоления порога вхождения. Как показывают наблюдения, отрицательное влияние этого рода более значимо, чем при преодолении порога вхождения в случае использования новых возможностей языка.

На частном уровне разработчик может самостоятельно добавить в процесс такие инструменты как аннотации типов в качестве промежуточной функции конвейера. Но при этом растут затраты на поиск способов оптимизации сценариев автоматизации.

Основным трендом изменения производительности серверных веб-приложении следует считать переход к фреймворкам, использующим экспериментальные возможности языка и строгую статическую типизацию. В этом случае такие затраты уже включены в стоимость освоения фреймворка.

Данное исследование направлено на практическое определение количественных характеристик факторов, косвенно и явно влияющих на время разработки и производительность конечного серверного приложения. Ввиду этого необходимо провести сравнение показателей фреймворков при использовании разных стандартов языка, экспериментальных особенностей, внедрения процесса сборки в единый файл.

Для того, чтобы унифицировать задачу определения производительности, в исследовании предлагается ввести общий шаблон приложения, которое будет реализовано на разных фреймворках с использованием различных возможностей языка, но предоставляющее единый функционал. Далее необходимо подвергнуть полученные приложения нагрузочному тестированию, косвенно определяющему производительность фреймворков и стека выбранных технологий.

Предварительно требуется отметить, что согласно исследованиям зарубежных авторов, определение производительности с помощью бенчмаркинга показывает 20%-й средний рост производительности серверных веб-приложений при переходе от фреймворков семейства Express к основанному на TypeScript фреймворку Nest. Эти данные предполагается подтвердить в ходе текущего анализа.

Выводы. Производительность серверного веб-приложения – это комплексное понятие, отражающее временные затраты на его разработку в неявной форме и на исполнение клиентских запросов в явной форме. Основным трендом её изменения на современном этапе является включение средств строгой типизации, транспиляции и сборки в состав фреймворков, что позволяет добиться более плавной кривой обучения и в конечном итоге более высокой отзывчивости и пропускной способности.

Проведенное исследование может быть использовано в качестве фундамента для первоначального выбора инструментария создания серверных приложений относительно возможностей и запросов разработчиков.

Демашов Д.С. (автор) Подпись

Государева И.Б. (научный руководитель) Подпись