Программные системы и вычислительные методы
Правильная ссылка на статью:
Смирнов А.А., Подольский Е.А., Черенков А.В., Государев И.Б. Сравнительный анализ производительности сред исполнения JavaScript кода: Node.js, Deno и Bun // Программные системы и вычислительные методы. 2024. № 4. DOI: 10.7256/2454-0714.2024.4.72206 EDN: KAZINO URL: https://nbpublish.com/library_read_article.php?id=72206

Сравнительный анализ производительности сред исполнения JavaScript кода: Node.js, Deno и Bun

Смирнов Андрей Александрович

ORCID: 0009-0006-6322-6842

студент; факультет "Программная инженерия и компьютерные технологии"; Национальный исследовательский университет ИТМО

197101, Россия, г. Санкт-Петербург, ул. Кронверкский, 49

smirnov.andrew.1999@yandex.ru

Подольский Егор Александрович

ORCID: 0009-0002-8311-8882

студент; факультет "Программная инженерия и компьютерные технологии"; Национальный исследовательский университет ИТМО

195112, Россия, г. Санкт-Петербург, ул. Заневский, 28-30-32, кв. 614

egorpodolskij51@yandex.ru

Черенков Артем Вячеславович

студент; факультет "Программная инженерия и компьютерные технологии"; Национальный исследовательский университет ИТМО

197101, Россия, г. Санкт-Петербург, ул. Кронверкский, 49 лит. А

art.cherenkov@gmail.com

Государев Илья Борисович

ORCID: 0000-0003-4236-5991

кандидат педагогических наук

доцент; факультет "Программная инженерия и компьютерные технологии"; Национальный исследовательский университет ИТМО

197046, Россия, г. Санкт-Петербург, Кронверкский пр-т, 49

goss@itmo.ru

Статья из рубрики "Операционные системы "

DOI:

10.7256/2454-0714.2024.4.72206

EDN:

KAZINO

Дата направления статьи в редакцию:

04-11-2024

Аннотация: Предметом исследования послужила производительность исполнения программ на JavaScript в современных средах Node.js, Deno и Bun. Эти платформы используются для разработки серверных приложений и имеют значительные различия в архитектуре, функциональных возможностях и производительности. Node.js — это наиболее зрелое и широко распространенное решение, которое активно используется в большинстве современных веб-приложений. Deno — это более новая среда, разработанная создателем Node.js, предлагающая улучшенную безопасность, поддержку TypeScript и другие нововведения. Bun, в свою очередь, представляет собой современную и высокопроизводительную альтернативу, ориентированную на скорость работы с серверными приложениями. Цель исследования заключается в выявлении различий в производительности основных современных сред исполнения (Node.js, Deno и Bun) для дальнейшего применения этих сред в разработке серверной части веб-приложений. Для проведения исследования использовался метод компьютерного эксперимента с применением Docker-контейнеров и автоматизации процессов с помощью Ansible. Измерялось время выполнения различных сценариев в каждой из сред исполнения. Научная новизна данного исследования заключается в том что впервые предложена целостная и обоснованная методика измерения и сравнения производительности кода на JavaScript применительно к современным средам исполнения, которая позволит исследователям в дальнейших экспериментах опираться на предложенный подход и распространить его на новые среды исполнения. Результаты исследования показывают, что Bun демонстрирует наилучшую производительность в синхронных вычислениях (сортировка, обработка JSON), но уступает Node.js и Deno в проверке на простоту числа. Deno проявил высокую эффективность в асинхронных операциях, благодаря использованию Rust и библиотеки Tokio. Node.js, несмотря на более низкие результаты в синхронных задачах, показал стабильную производительность в тестах и остается надежным выбором для крупных проектов. В ходе исследования были разработаны рекомендации по выбору подходящей среды выполнения серверного JavaScript кода для различных задач.

Ключевые слова:

JavaScript, Node.js, Deno, Bun, Производительность, Компьютерный эксперимент, Бэкенд, Веб, Сервер, Docker

Введение

JavaScript был анонсирован в 1996 году. Целью языка было добавление интерактивности на статичные веб-страницы. С этого момента прошло много времени. JavaScript стал одним из наиболее популярных языков программирования, а область его применения расширилась. Ключевым расширением функциональности языка стало появление среды

исполнения серверного JavaScript кода - Node.js в 2009 году. Node.js позволила использовать один язык как для разработки клиента, так и для создания сервера. Это привело к активному развитию технологий на базе JavaScript, таких как серверные приложения, микросервисы и API.

Несовершенство Node.js и востребованность в средах выполнения серверного JavaScript кода подтолкнули индустрию к развитию и созданию новых решений. Создатель Node.js Райан Даль в 2018 представил новую технологию - Deno, которая должна была решить все проблемы предшественника. Вслед на Deno в 2022 году была анонсированная новая среда - Bun. Каждая из представленных технологий предлагает уникальные подходы к безопасности, производительности и удобству разработки. Однако, несмотря на растущее число исследований и статей о сравнении сред выполнения JavaScript, до сих пор не проводилось комплексного анализа всех трех технологий, учитывающего все ключевые критерии.

Объектом данного исследования является язык JavaScript и исполнение программного кода на этом языке.

Предметом исследования послужила производительность исполнения программ на JavaScript в современных средах.

Цель исследования – выявить различия в производительности основных современных сред исполнения (Node.js, Deno и Bun) для дальнейшего применения этих сред в разработке серверной части веб-приложений.

Научная новизна данного исследования заключается в том что впервые предложена целостная и обоснованная методика измерения и сравнения производительности кода на JavaScript применительно к современным средам исполнения, которая позволит исследователям в дальнейших экспериментах опираться на предложенный подход и распространить его на новые среды исполнения.

Перспективы

Область программных систем на веб-платформе развивается крайне динамично и сопровождается разработкой новых сред исполнения. До настоящего времени эти среды разрабатывались за рубежом. В русле импортозамещения планируется выработать систему критериев оценки производительности исполнения JavaScript-кода и использовать её для разработки отечественных решений в данной сфере. В числе прочего дальнейшие исследования будут направлены на разработку российских компонентов облачных инфраструктур, обеспечивающих безопасные и эффективные методы поддержки серверных решений, основанных на экосистеме JavaScript.

Анализ литературы

В научной литературе тема сравнения производительности сред исполнения JavaScript кода исследована недостаточно. Большинство работ фокусируются на сравнении производительности различных языков программирования или сравнивают только две среды исполнения [1, 2]. Официальные сайты Node.js, Deno и Bun заявляют о превосходстве своих продуктов, но часто без подробных экспериментальных данных.

В работе [3] проведено сравнение Deno и Bun, где показано, что Bun выделяется в обработке массивов и рекурсивных вычислениях, а Deno — в асинхронных задачах. В статье [4] проведено сравнение Node.js и Bun, где отмечается, что Bun обещает

значительно более быстрый запуск, что делает его привлекательным для бессерверных функций.

В статье [5] отмечается, что хотя Bun все еще нов и требуется больше всесторонних испытаний, начальные тесты показывают многообещающие результаты. Легкая модель корутин и оптимизированный сборщик мусора в Bun могут обеспечить улучшенную производительность для нагрузки, связанной с вводом-выводом. Однако, относительная новизна Bun и меньшее сообщество по сравнению с Node.js могут создать препятствия для его широкого применения.

В работе [6] проведено сравнение производительности Node.js и Deno, показывающее приблизительно равный уровень производительности в задачах преобразования координат и поиска прямых линий в изображениях. Это связано с использованием одного и того же движка V8.

В книге [7] описываются основные требования к профессиональному бенчмаркингу: повторяемость, проверяемость и переносимость, принцип невмешательства, приемлемый уровень точности и честность. Авторы также рассматривают риски бенчмаркинга, такие как неточные замеры времени и оптимизация компилятора. Все эти требования и рекомендации учтены в данном исследовании.

В статье [8] рекомендуется использовать график "ящик с усами" для наглядного представления распределения данных. Этот график отображает распределение данных через их квартили и выявляет потенциальные выбросы, что особенно полезно для сравнения распределений между различными группами или условиями.

Исследуемая в работе [9] модель M/G/m queue основывается на том, что современные веб-серверы обладают очередью запросов и пулом потоков для обработки входящих запросов. В средах исполнения JavaScript вместо пула потоков используется механизм Event Loop, который обеспечивает асинхронность выполнения операций. Синхронные операции, однако, выполняются в основном потоке, который существует в единственном экземпляре. Если этот поток будет занят выполнением ресурсоемких вычислений, то время ожидания ответа кратно увеличится.

В статье [10] проводится экспериментальное исследование с целью измерения и сравнения производительности Node.js и Jetty в контексте обработки REST-запросов. В рамках данного исследования сервер выполняет ряд операций, характерных для каждого запроса: парсинг JSON, создание объектов, их деструктуризация, обработка, сбор данных и формирование ответа в формате JSON. Эти операции, являющиеся неотъемлемой частью обработки каждого запроса, были выбраны для анализа производительности в данной работе.

Методология эксперимента

Для обеспечения воспроизводимости и переносимости экспериментальных вычислений все операции выполнялись в изолированной среде Docker-контейнеров. Создание Docker-образов осуществлялось на базе стандартных образов, доступных в репозитории Docker Hub (таблица 1). В качестве основы были выбраны Alpine-образы, которые отличаются минималистичностью и повышенной безопасностью.

Таблица 1 – Используемые базовые Docker-образы

Node.js	Deno	Bun

Node.js	Deno	Bun
node:21-alpine	denoland/deno:alpine-1.42.0	oven/bun:1.1.2-alpine

Исследования проводились в облачных вычислительных средах с использованием виртуальных машин (таблица 2). Такой подход позволяет приблизить условия эксперимента к реальным производственным условиям.

Таблица 2 – Характеристики виртуальной машины

Операционная система	Процессор	ОЗУ
ubuntu-22.04-live-server-amd64.iso	2x2.2 ГГц	2 Гб

Описание процесса эксперимента представлено на рисунке 1.

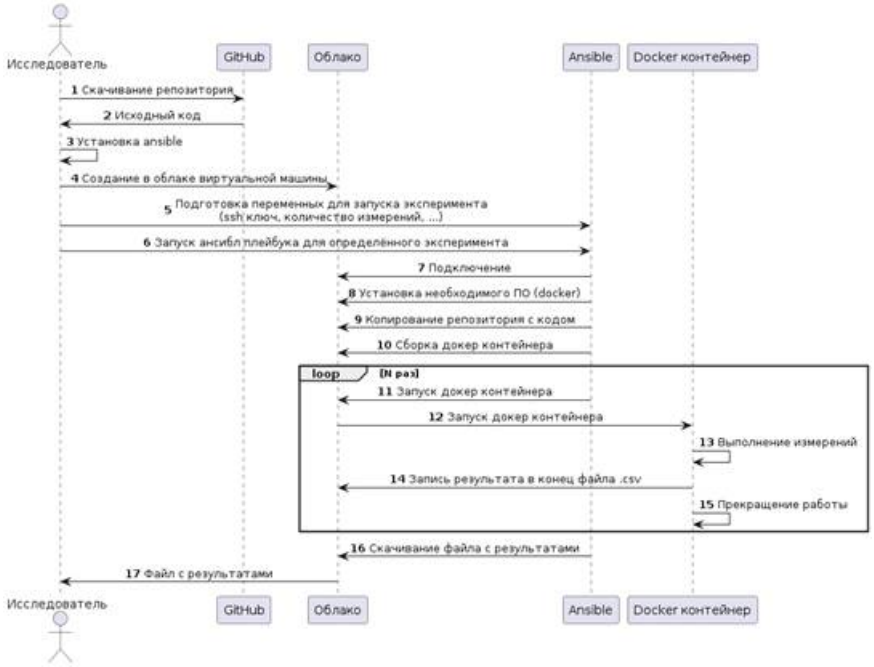

Рисунок 1 – Процесс проведения эксперимента

Эксперимент включал измерение времени выполнения следующих сценариев каждый из которых запускался 1000 раз для обеспечения достаточного количество элементов в выборке:

- Быстрая сортировка. Быстрая сортировка массива из 10000 чисел, сгенерированных с помощью функции Math.random().
- Обработка JSON. JSON.stringify, а затем JSON.parse объекта с 10000 ключами без вложенностей.
- Деструктуризация. Создание нового объекта из двух объектов, у каждого из которых по 10000 ключей, с помощью деструктуризации.
- Object.assign. Создание нового объекта из двух объектов, у каждого из которых по 10000 ключей, с помощью Object.assign.
- Проверка на простоту числа. Является ли число 5600748293801 простым.
- Обработка промисов. Создание и await пустого промиса в цикле 1 млн раз.

Результаты эксперимента

По результатам эксперимента составлены сравнительные таблицы с основными

характеристиками распределения и построены графики. В ходе эксперимента было зафиксировано, что средняя загрузка двухъядерного центрального процессора составила 58%. Это явление можно объяснить следующим образом: одно из ядер процессора испытывало максимальную нагрузку в связи с выполнением задач однопоточных движков, в то время как другое ядро обеспечивало выполнение фоновых процессов операционной системы.

В ходе эксперимента наблюдалось постепенное увеличение объема оперативной памяти, используемой для выполнения задач, с 400 Мб до 500 Мб. Выделение 2 Гб виртуальной памяти оказалось избыточным для выполнения задач.

Deno показал наилучшую производительность при выполнении быстрой сортировки массива, с наименьшим средним временем выполнения 17,3 мс и минимальным значением 11,5 мс. Bun занял второе место со средним временем 19,1 мс, а минимальным значением 13,2 мс. Node.js показал наихудшие результаты со средним временем 23,5 мс. Числовые показатели представлены в таблице 3, а их визуализация на рисунке 3.

Таблица 3 – Результаты быстрой сортировки массива

Время, мс	**Node.js**	**Deno**	**Bun**
Среднее	23,5	17,3	19,1
Медиана	21,5	16,8	18,3
Максимальное	51,4	42,0	33,5
Минимальное	16,3	11,5	13,2
Среднеквадратичное отклонение	4,6	2,7	3,0

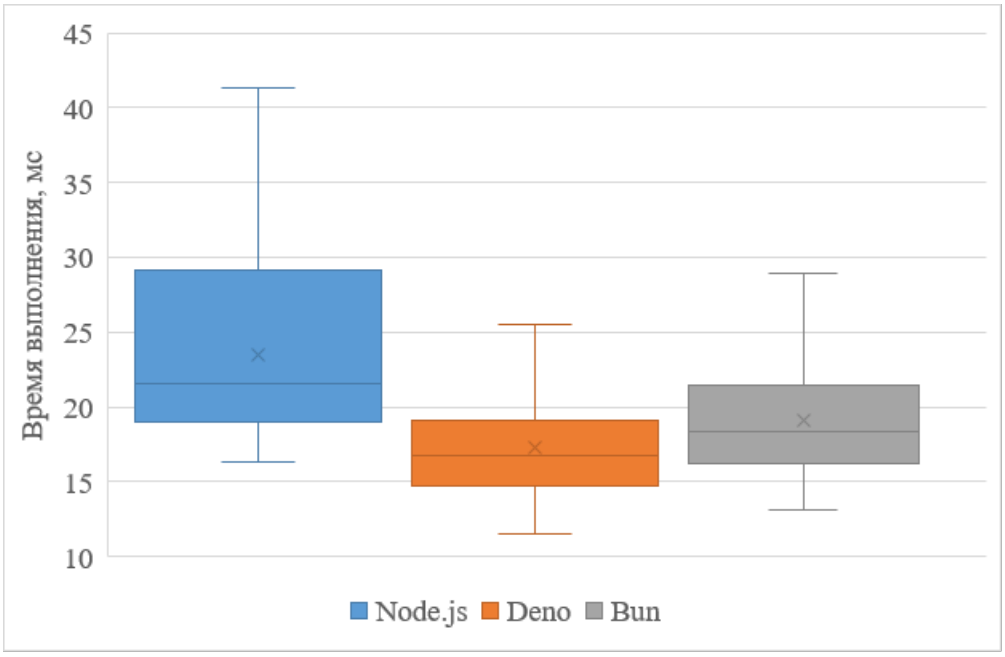

Рисунок 2 – Ящик с усами для результатов быстрой сортировки

Большое, относительно других экспериментов, среднеквадратичное отклонение объясняется тем, что время выполнения алгоритма быстрой сортировки зависит от того, как в массиве распределены элементы, которые в данном эксперименте генерировались случайным образом.

Для повышения наглядности масштаба основного распределения выбросы не отображены на графических иллюстрациях, поскольку их присутствие затрудняет визуальное восприятие распределения данных.

Bun продемонстрировал наилучшую производительность в обработке JSON, со средним временем выполнения 5,8 мс и минимальным значением 5,3 мс. Deno занял второе место со средним временем 13,4 мс, а Node.js показал наихудшие результаты со средним временем 15,0 мс. Среднеквадратичное отклонение было самым низким у Bun (0,3 мс), что указывает на высокую стабильность его производительности. Максимальное время больше всего отличается от медианы (на 56%) у Node.js, что свидетельствует о значительном выбросе. Числовые показатели представлены в таблице 4, а их визуализация на рисунке 4.

Таблица 4 – Результаты обработки JSON

Время, мс	**Node.js**	**Deno**	**Bun**
Среднее	15,0	13,4	5,8
Медиана	14,5	13,1	5,7
Максимальное	22,6	15,1	7,3
Минимальное	14,0	12,8	5,3
Среднеквадратичное отклонение	0,9	0,5	0,3

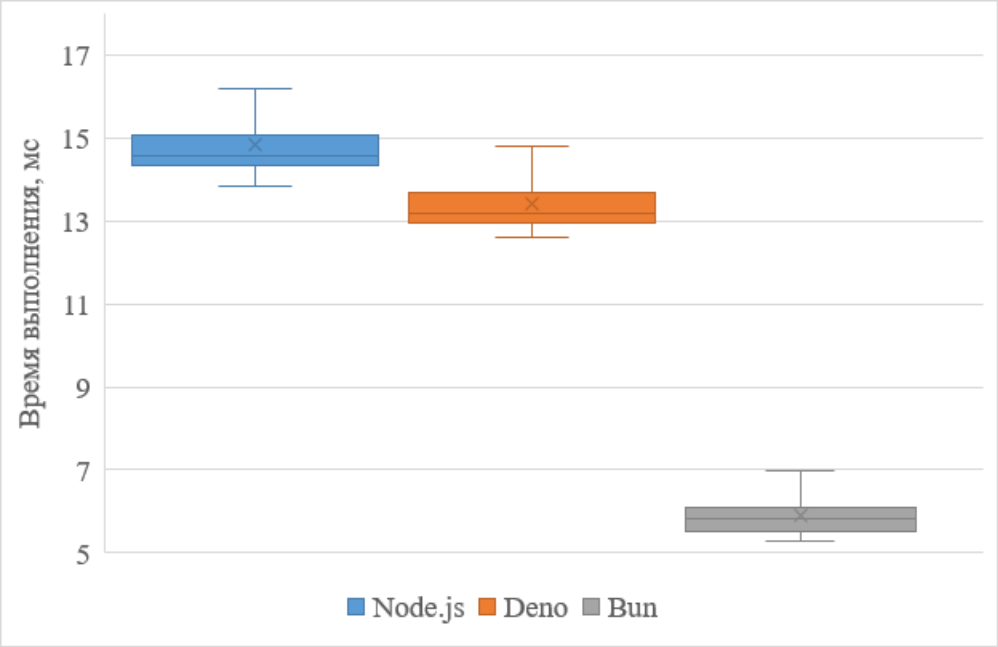

Рисунок 3 – Ящик с усами для результатов обработки JSON

Bun значительно превосходит Deno и Node.js в выполнении операций деструктуризации, со средним временем 3,5 мс и минимальным значением 3,1 мс. Deno занял второе место со средним временем 15,5 мс, а Node.js показал наихудшие результаты со средним временем 17,0 мс. Среднеквадратичное отклонение у Bun было самым низким (0,2 мс), что свидетельствует о стабильной производительности. Числовые показатели представлены в таблице 5, а их визуализация на рисунке 5.

Таблица 5 – Результаты деструктуризации

Время, мс	Node.js	Deno	Bun
Среднее	17,0	15,5	3,5
Медиана	16,7	15,3	3,4
Максимальное	45,2	33,1	7,1
Минимальное	15,8	14,2	3,1
Среднеквадратичное отклонение	0,7	0,7	0,2

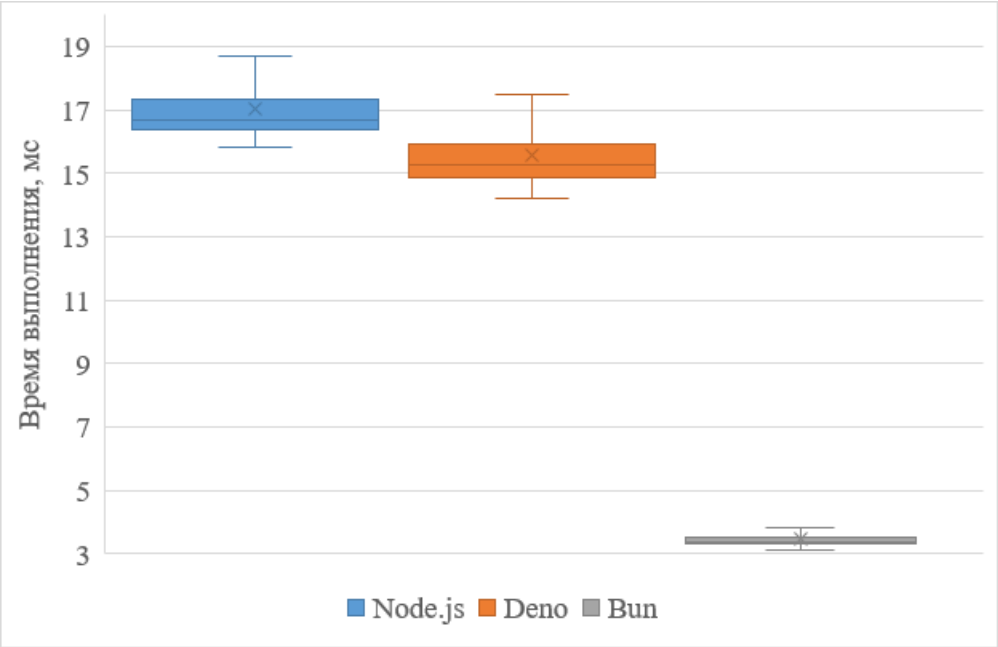

Рисунок 4 – Ящик с усами для результатов деструктуризации

Bun продемонстрировал наилучшую производительность в операции Object.assign, со средним временем 3,3 мс и минимальным значением 3,0 мс. Deno показал среднее время выполнения 16,9 мс, а Node.js – 18,3 мс. Среднеквадратичное отклонение у Bun было самым низким (0,2 мс), что указывает на стабильную производительность. Числовые показатели представлены в таблице 6, а их визуализация на рисунке 6.

Таблица 6 – Результаты Object.assign

Время, мс	Node.js	Deno	Bun
Среднее	18,3	16,9	3,3
Медиана	18,1	16,6	3,2
Максимальное	44,0	32,7	11,1
Минимальное	17,0	15,8	3,0
Среднеквадратичное отклонение	0,6	0,7	0,2

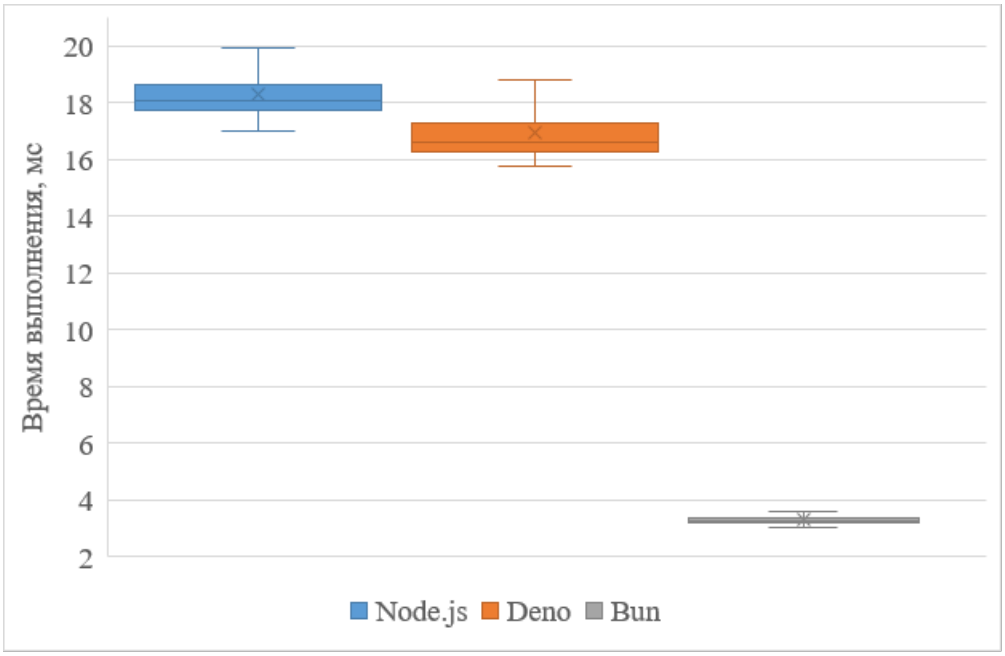

Рисунок 5 – Ящик с усами для результатов Object.assign

Node.js и Deno показали схожие результаты при проверке числа на простоту, со средними значениями 16,7 мс и 16,0 мс соответственно. Bun продемонстрировал значительно худшие результаты, со средним временем выполнения 119,5 мс. Среднеквадратичное отклонение у Bun было самым высоким (2,7 мс), что свидетельствует о меньшей стабильности. Числовые показатели представлены в таблице 7, а их визуализация на рисунке 7.

Таблица 7 – Результаты проверки на простоту числа

Время, мс	**Node.js**	**Deno**	**Bun**
Среднее	16,7	16,0	119,5
Медиана	16,3	15,7	119,7
Максимальное	50,8	27,0	150,8
Минимальное	15,7	14,9	113,1
Среднеквадратичное отклонение	0,8	0,8	2,7

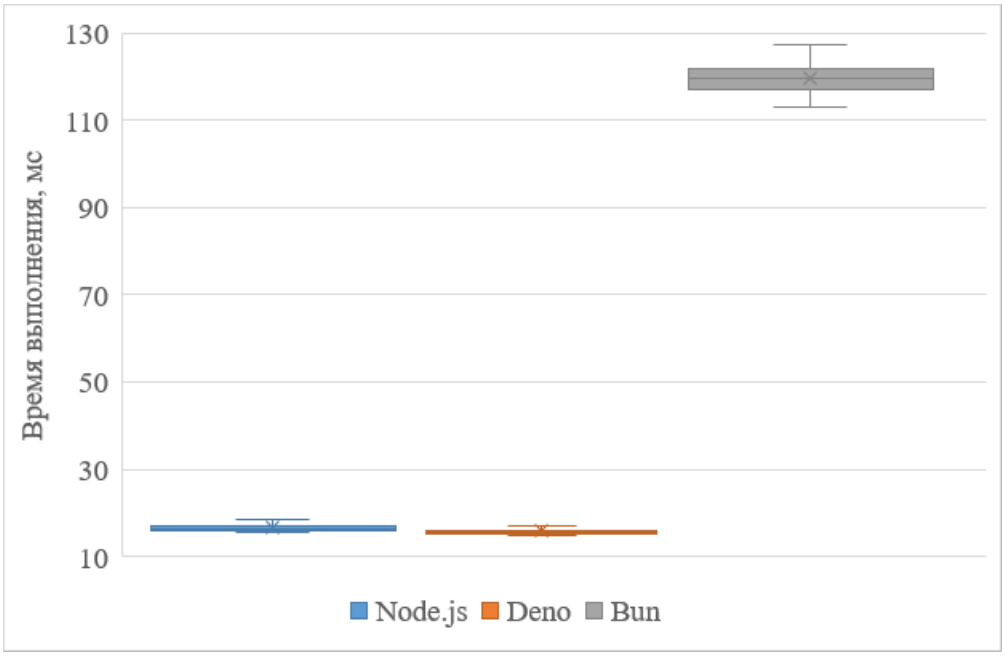

Рисунок 6 – Ящик с усами для результатов проверки на простоту числа

Таблица 8 – Результаты обработки промисов

Время, мс	Node.js	Deno	Bun
Среднее	195,5	173,1	230,1
Медиана	194,9	172,3	229,4
Максимальное	233,4	193,0	257,3
Минимальное	183,7	162,8	214,5
Среднеквадратичное отклонение	3,9	4,0	5,7

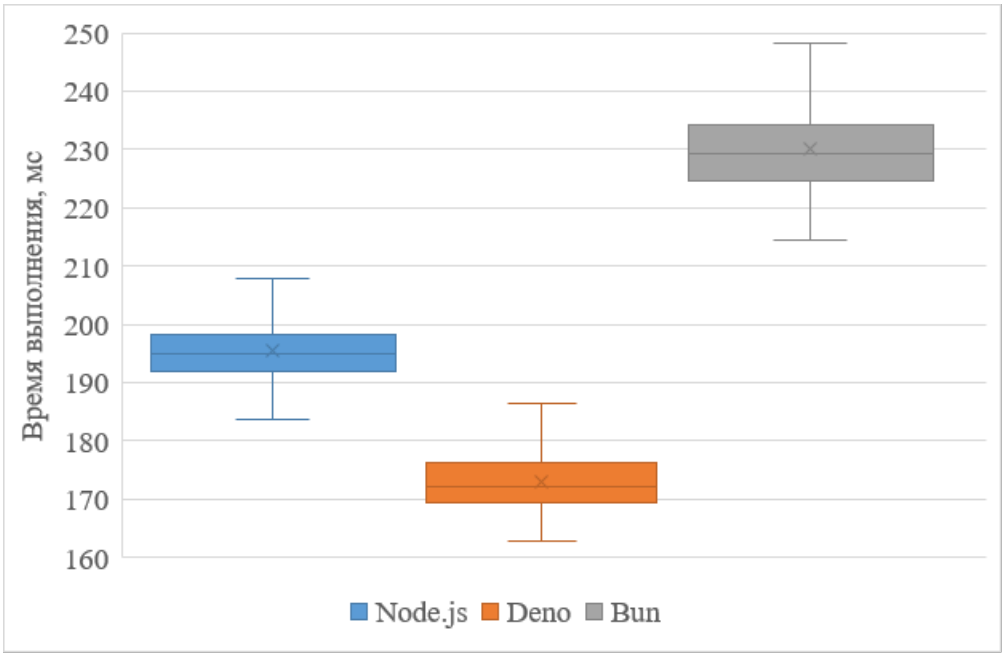

Рисунок 7 – Ящик с усами для результатов обработки промисов

Deno показал наилучшую производительность при обработке промисов, со средним временем выполнения 173,1 мс и минимальным значением 162,8 мс. Node.js занял второе место со средним временем 195,5 мс, а Bun показал наихудшие результаты со средним временем 230,1 мс.

Заключение

В данном исследовании проведён комплексный анализ производительности сред исполнения JavaScript кода: Node.js, Deno и Bun в различных сценариях, включая быструю сортировку, обработку JSON, деструктуризацию, Object.assign, проверку на простоту числа и обработку промисов. Эксперимент был выполнен в изолированной среде Docker-контейнеров с использованием инструмента Ansible для автоматизации процесса, что обеспечило воспроизводимость и переносимость результатов.

Результаты эксперимента показали, что Bun демонстрирует наилучшую производительность в большинстве синхронных вычислений, таких как быстрая сортировка, обработка JSON, деструктуризация и Object.assign. Однако, в сценарии проверки на простоту числа Bun значительно уступает Node.js и Deno, что может быть обусловлено особенностями реализации расчёта остатка от деления в Bun. Deno, в свою очередь, демонстрировал высокую эффективность в асинхронных операциях, таких как обработка промисов, благодаря тому, что он разработан на языке Rust и использует библиотеку Tokio для асинхронных задач, что способствовало его оптимизированной асинхронной архитектуре.

Node.js, несмотря на то, что он не выигрывает в большинстве синхронных сценариев, обеспечивает стабильную производительность во всех тестах и остается надежным выбором для крупных проектов, где важна проверенная временем стабильность и наличие специализированных библиотек. Более того, широкое распространение и большее количество разработчиков, знакомых с Node.js, делает его предпочтительным в тех случаях, когда требуется общая совместимость и поддержка сообщества.

Рекомендуется использовать Bun для задач, требующих высокой скорости сборки и эффективности в синхронных вычислениях, особенно в бессервисных функциях. Однако, при выборе Bun для задач, связанных с асинхронными операциями или специфическими вычислениями, следует проводить дополнительные исследования производительности, так как в некоторых сценариях он может оказаться менее эффективным, чем Node.js или Deno.

Deno, благодаря своей высокой производительности в асинхронных операциях и акценту на безопасности, является предпочтительным выбором для задач, где важен баланс между безопасностью и производительностью. Кроме того, Deno показал себя более быстрым, чем Node.js, во всех сценариях.

Предполагается, что похожие результаты производительности между Node.js и Deno могут быть обусловлены тем, что обе среды исполнения используют одинаковый движок V8, что подтверждает важность учёта движка при оценке производительности JavaScript сред.

В таблице 9 приведены относительные временные характеристики вычислений медиан, где минимальное время принято за 100%, для удобного сравнения результатов между собой.

Таблица 9 – Относительные результаты вычислений для медиан

Название эксперимента	Node.js	Deno	Bun
Быстрая сортировки массива	128%	100%	109%
Обработка JSON	254%	230%	100%
Деструктуризация	491%	450%	100%
Object.assign	566%	519%	100%
Проверка на простоту числа	104%	100%	762%
Обработки промисов	113%	100%	133%

В целом, выбор среды исполнения JavaScript должен основываться на конкретных требованиях проекта, таких как тип вычислений (синхронные или асинхронные), необходимость безопасности, стабильность. Результаты данного исследования предоставляют разработчикам обоснованные рекомендации для выбора наиболее подходящей среды в зависимости от специфики задачи.

Библиография

1. Martinsen, J., Grahn, H. A methodology for evaluating JavaScript execution behavior in interactive web applications. // В материалах 9-й Международной конференции по компьютерным системам и приложениям IEEE/ACS (AICCSA), 2011. С. 241-248. DOI: 10.1109/AICCSA.2011.6126611.
2. Родыгина И. В., Наливайко А. В. Сравнительный анализ технологий для разработки серверной части системы управления продажами: статья в журнале // Известия ЮФУ. Технические науки. № 4 (221). 2021. С. 256-266. DOI: 10.18522/2311-3103-2021-4-256-266.
3. Тихонов Д.С., Черенков А.В., Долгов А.В. Сравнительный анализ технологий серверной разработки на платформах deno и bun // Научно-технические инновации и веб-технологии. 2023. № 2. С. 55-59. URL: https://elibrary.ru/item.asp?Id=56409551.
4. Kniazev I., Fitiskin A. Choosing the right javascript runtime: an in-depth comparison of node.js and bun: статья в журнале // Norwegian journal of development of the international science. 2023. С. 72-84. DOI: 10.5281/zenodo.7945166. URL: https://elibrary.ru/item.asp?Id=53844306.
5. Koper D., Woda M. Performance Analysis and Comparison of Acceleration Methods in javascript Environments Based on Simplified Standard Hough Transform Algorithm //International Conference on Dependability and Complex Systems. Cham: Springer International Publishing, 2022. С. 131-142. DOI: 10.1007/978-3-031-06746-4_13.
6. Акиньшин А. Профессиональный бенчмарк: искусство измерения производительности. спб.: Питер, 2022. 576 с.: ил. (Серия «Библиотека программиста»). ISBN 978-5-4461-1551-8.
7. Сальникова, К. В. Анализа массива данных с помощью инструмента визуализации "Ящик с усами" / К. В. Сальникова // Universum: экономика и юриспруденция. 2021. № 6(81). С. 11-17. DOI 10.32743/unilaw.2021.81.6.11778.
8. Gregg B. BPF Performance Tools (Addison-Wesley Professional Computing Series) // Update. Т. 517. С. 18. 2019.
9. Lu J., Gokhale S. S. performance analysis of a Web Server //International Journal of Information Technology and Web Engineering (IJITWE). – 2008. – Т. 3. – № 3. – С. 50-65.
10. Lundar J. A., Grønli T. M., Ghinea G. Performance evaluation of a modern web

architecture // International Journal of Information Technology and Web Engineering (IJITWE). – 2013. – Т. 8. – № 1. – С. 36-50.
11. Смирнов А. А., Черенков А. В., Подольский Е. А. Сравнительный анализ сред исполнения Javascript кода Node.js, Deno и Bun для разработки серверной части веб-приложения // Международный журнал информационных технологий и энергоэффективности. 2024. Т. 9, № 4(42). С. 100-106.
12. Lei, K., Ma, Y., & Tan, Z. (2014). Performance Comparison and Evaluation of Web Development Technologies in PHP, Python, and Node.js. // В материалах 17-й Международной конференции IEEE по вычислительной науке и инженерии (CSE), 2014. С. 661-668. DOI: 10.1109/CSE.2014.142.
13. Суворов Д. А. Измерение текущего быстродействия процессоров и качества программ. Способы оценки и повышения реальной производительности //Открытое образование. 2009. № 6. С. 59-65.
14. Пентковский В. М. и др. Методология оценки производительности вычислительных систем //Актуальные проблемы современной науки. 2012. № 6. С. 358-363.
15. Жуйков Р., Шарыгин Е. Методы предварительной оптимизации программ на языке JavaScript //Труды Института системного программирования РАН. 2015. Т. 27. № 6. С. 67-86.
16. B. Basumatary, N. Agnihotri BENEFITS AND CHALLENGES OF USING NODE.JS // International Journal of Innovative Research in Computer Science & Technology. 2022. № 10-3. С. 67-70. URL: https://acspublisher.com/journals/index.php/ijircst/article/view/10433/9623

Результаты процедуры рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.
Со списком рецензентов издательства можно ознакомиться здесь.

Представленная статья на тему «Сравнительный анализ производительности сред исполнения JavaScript кода: Node.js, Deno и Bun» соответствует тематике журнала «ПРОГРАММНЫЕ СИСТЕМЫ И ВЫЧИСЛИТЕЛЬНЫЕ МЕТОДЫ» и посвящена актуальн вопросу, связанного с ростом популярности JavaScript и его применения в веб-разработке, чтоприводит к выбору оптимальной среды исполнения. JavaScript, изначально предназначенный для динамического взаимодействия на веб-страницах имеет широкое использование в серверных приложениях, микросервисах и других высоконагруженных системах. В таких условиях разработчики сталкиваются с необходимостью оценить производительность различных сред исполнения, таких как Node.js, Deno и Bun, чтобы выбрать наиболее подходящую для конкретных задач.
В статье представлен достаточно широкий анализ литературных российских и зарубежных источников по вопросам сравнения сравнение Deno и Bun, сравнения производительности Node.js и Deno, экспериментального исследования с целью измерения и сравнения производительности Node.js и Jetty в контексте обработки REST-запросов.
Авторами самостоятельно проведен комплексный анализ производительности Node.js, Deno и Bun в различных сценариях исполнения JavaScript кода и др. параметров.
Стиль и язык изложения материала является достаточно доступным для широкого круга читателей.
Практическая значимость статьи четко обоснована. Статья по объему соответствует рекомендуемому объему от 12 000 знаков.
Статья достаточно структурирована - в наличии введение, заключение, внутреннее

членение основной части (анализ литературы, методология эксперимента, результаты эксперимента).

Авторами проведен эксперимент по воспроизводимости и переносимости вычислений, выполненных в изолированной среде Docker-контейнеров. Создание Docker-образов осуществлялось на базе стандартных образов, доступных в репозитории Docker Hub, результаты обобщены в виде табдлицы. В качестве основы были выбраны Alpine-образы, которые отличаются минималистичностью и повышенной безопасностью.

Будет целесообразным добавить в водную часть статьи краткое рассмотрение истории вопроса применения JavaScript для подведения к современному положению дел и обоснованию актуальности рассматриваемой проблемы.

К недостаткам можно отнести следующие моменты: из содержания статьи не прослеживается научная новизна. Отсутствует четкое выделение предмета, объекта и цели исследования, сжатые выводы.

Рекомендуется четко обозначить научную новизну исследования, сформулировать предмет, объект, цель исследования и развернуто написать выводы (содержащие по необходимости рекомендации, оценки, предложения, гипотезы и др.). Также будет целесообразным добавить о перспективах дальнейшего исследования.

Статья «Сравнительный анализ производительности сред исполнения JavaScript кода: Node.js, Deno и Bun» требует доработки по указанным выше замечаниям. После внесения поправок рекомендуется к повторному рассмотрению редакцией рецензируемого научного журнала.

Результаты процедуры повторного рецензирования статьи

В связи с политикой двойного слепого рецензирования личность рецензента не раскрывается.

Со списком рецензентов издательства можно ознакомиться здесь.

Рецензируемая статья посвящена сравнительному анализу производительности различных сред исполнения JavaScript кода, предметом исследования в работе выступает производительность исполнения программ в современных средах: Node.js, Deno и Bun.

Методология исследования базируется на проведении экспериментальных вычислений в изолированной среде Docker-контейнеров в облачных вычислительных средах с использованием виртуальных машин, а также обобщении сведений из ранее опубликованных работ , посвященных решаемой проблеме.

Актуальность работы авторы связывают с тем, что JavaScript стал одним из наиболее популярных языков программирования, существующие среды исполнения серверного JavaScript кода: Node.js, Deno и Bun предлагают уникальные подходы к безопасности, производительности и удобству разработки, однако до сих пор не проводилось комплексного анализа всех трех технологий, учитывающего все ключевые критерии.

Научная новизна рецензируемого исследования состоит в том, что авторами предложена целостная и обоснованная методика измерения и сравнения производительности кода на JavaScript применительно к современным средам исполнения, которая позволит исследователям в дальнейших экспериментах опираться на предложенный подход и распространить его на новые среды исполнения.

В статье структурно выделены следующие разделы: Введение, Перспективы, Анализ литературы, Методология эксперимента, Результаты эксперимента, Заключение и Библиография.

Авторами приведен обзор литературы по теме публикации; детально отражен ход проведения эксперимента, описаны используемые базовые Docker-образы, приведены

характеристики виртуальной машины, представлена схема процесса проведения эксперимента; показаны результаты быстрой сортировки массива, обработки JSON, деструктуризации, операции Object.assign, проверки на простоту числа и обработки промисов в трёх средах. В Заключении сказано, что что Bun демонстрирует наилучшую производительность в большинстве синхронных вычислений, таких как быстрая сортировка, обработка JSON, деструктуризация и Object.assign; Node.js обеспечивает стабильную производительность во всех тестах и остается надежным выбором для крупных проектов, где важна проверенная временем стабильность и наличие специализированных библиотек; Deno, благодаря своей высокой производительности в асинхронных операциях и акценту на безопасности, является предпочтительным выбором для задач, где важен баланс между безопасностью и производительностью. Полученные результаты исследования предоставляют разработчикам обоснованные рекомендации для выбора наиболее подходящей среды в зависимости от специфики задачи.

Библиографический список включает 16 источников – публикации отечественных и зарубежных ученых на русском и английском языках по теме статьи, на которые в тексте имеются адресные ссылки, подтверждающие наличие апелляции к оппонентам.

Статья отражает результаты проведенного авторами исследования, соответствует направлению журнала «Программные системы и вычислительные методы», содержит элементы научной новизны и практической значимости, может вызвать интерес у читателей, рекомендуется к опубликованию.