АРХИТЕКТУРНЫЕ ПОДХОДЫ К ОБРАБОТКЕ АСИНХРОННЫХ ОПЕРАЦИЙ В СРЕДАХ ВЫПОЛНЕНИЯ NODE.JS И BUN

Короткая М.Д., студент

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Мелких Е.Г. – старший преподаватель

В работе рассматриваются архитектурные особенности обработки асинхронных операций в средах выполнения Node.js и Bun. Выполнен сравнительный анализ моделей событийного цикла, механизмов работы с промисами и низкоуровневых оптимизаций. Показаны различия в реализации системного ввода-вывода, планирования задач и управления памятью. Полученные результаты позволяют выявить преимущества Bun в контексте производительности и Node.js – в зрелости экосистемы и стабильности.

Современные серверные среды выполнения JavaScript широко используют асинхронные модели для обеспечения высокой производительности и масштабируемости. Ключевыми представителями являются Node.js и Bun, каждая из которых характеризуется собственными архитектурными подходами к обработке асинхронных операций.

Node.js базируется на движке V8 и библиотеке libuv, обеспечивающей кроссплатформенную реализацию неблокирующего ввода-вывода. Центральным элементом архитектуры является событийный цикл, который последовательно обрабатывает очереди задач и функций обратного вызова, разделенных на фазы (таймеры, обработка ввода-вывода, ожидание событий, завершающие операции и др.). Такая модель обеспечивает высокую масштабируемость, однако наличие промежуточного слоя в виде libuv приводит к дополнительным накладным расходам при выполнении системных операций [1].

Более детальное рассмотрение внутреннего устройства Node.js показывает, что ключевую роль в обеспечении асинхронности играет библиотека libuv, реализующая неблокирующий ввод-вывод и управление пулом потоков. Несмотря на однопоточную модель выполнения JavaScript-кода, Node.js использует пул рабочих потоков для выполнения потенциально блокирующих операций, таких как доступ к файловой системе, криптографические вычисления и сетевые запросы. Это позволяет избежать блокировки основного потока выполнения.

Существенной особенностью архитектуры является разделение задач на макрозадачи и микрозадачи. Макрозадачи (например, обработка событий ввода-вывода) выполняются в рамках фаз событийного цикла, тогда как микрозадачи (например, обработка обещаний) выполняются с более высоким приоритетом между итерациями цикла. Однако такая модель требует дополнительных проверок очереди микрозадач после каждой фазы, что может приводить к увеличению накладных расходов при большом количестве коротких асинхронных операций.

Кроме того, взаимодействие между движком V8 и библиотекой libuv осуществляется через слой привязок на языке C++, что добавляет дополнительный уровень абстракции. При интенсивной нагрузке это может приводить к увеличению времени выполнения операций за счет дополнительных переходов между уровнями системы.

Схема архитектуры Node.js представлена на рисунке 1.

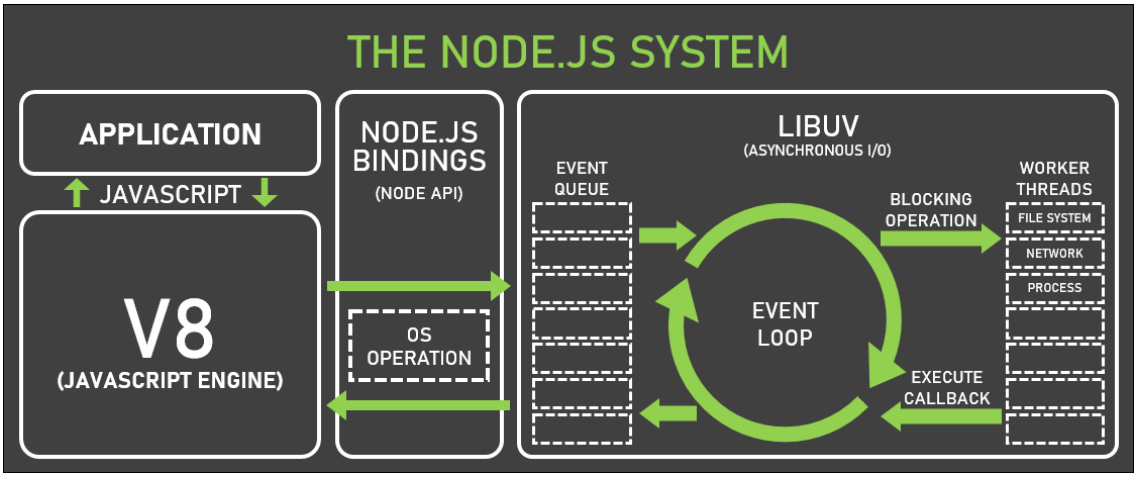

Рисунок 1 – Архитектура Node.js

В отличие от Node.js, среда Bun ориентирована на минимизацию накладных расходов и повышение производительности. Она построена на базе движка JavaScriptCore и реализует

значительную часть функциональности на уровне среды выполнения, включая обработку HTTP-запросов и работу с файловой системой. Это позволяет сократить количество переходов между уровнями абстракции и уменьшить задержки при выполнении операций.

Дополнительным фактором высокой производительности Bun является использование движка JavaScriptCore, который отличается иной архитектурой оптимизации выполнения кода и управления памятью по сравнению с V8. В частности, применяются более агрессивные методы оптимизации часто выполняемых участков кода, что позволяет сократить время выполнения повторяющихся операций.

Важной особенностью Bun является встроенная реализация ряда стандартных механизмов, которые в Node.js предоставляются внешними модулями. Например, средства работы с HTTP и файловой системой интегрированы непосредственно в среду выполнения. Это позволяет избежать дополнительных затрат на загрузку модулей и взаимодействие между уровнями абстракции.

Также Bun использует оптимизированные механизмы работы с сетевыми соединениями, позволяющие эффективнее обрабатывать большое количество одновременных запросов. За счет сокращения количества промежуточных структур данных и более эффективного управления буферами снижается нагрузка на систему памяти и уменьшаются задержки при передаче данных.

Дополнительной особенностью является оптимизация запуска приложений. В отличие от Node.js, где запуск может сопровождаться загрузкой значительного числа зависимостей, Bun использует более быстрые механизмы инициализации модулей и кэширования, что сокращает время старта.

Сравнительный анализ сред выполнения Node.js и Bun выявил принципиальные различия, обусловленные особенностями их архитектурной реализации. Среда Node.js отличается высокой степенью совместимости и стабильности, что связано с её зрелой экосистемой, широким распространением и наличием большого количества проверенных библиотек.

В то же время более высокая производительность Bun обусловлена рядом архитектурных оптимизаций на уровне среды выполнения. В отличие от Node.js, где значительная часть асинхронных операций проходит через слой libuv, Bun реализует многие механизмы напрямую с использованием системных программных интерфейсов. Например, обработка HTTP-запросов в Bun встроена в среду выполнения и не требует использования внешних модулей, что уменьшает количество промежуточных вызовов и снижает накладные расходы. Аналогично, операции чтения файлов выполняются с меньшим количеством уровней абстракции, что сокращает время выполнения системных вызовов.

Дополнительным фактором является уменьшение числа переключений контекста между JavaScript-движком и нативным уровнем. В Node.js выполнение асинхронной операции включает последовательность переходов: JavaScript, V8, C++ bindings, libuv, ОС и обратно. В Bun эта цепочка короче за счет более тесной интеграции с JavaScriptCore, что снижает задержки при интенсивных операциях ввода-вывода.

Оптимизация обработки асинхронных задач в Bun также достигается за счет эффективного управления очередями макрозадач и микрозадач. В Node.js после каждой фазы событийного цикла выполняется проверка очереди микрозадач, тогда как в Bun используется стратегия пакетной обработки, позволяющая выполнять несколько микрозадач подряд без возврата в основной цикл. Это снижает задержки при выполнении длинных цепочек асинхронных операций.

Повышенная эффективность обработки обещаний в Bun связана с особенностями реализации микрозадач в JavaScriptCore. Используется более легковесная модель управления, при которой операции создания и завершения обещаний требуют меньшего количества внутренних выделений памяти.

Дополнительно следует отметить различия в подходах к масштабированию приложений. В Node.js горизонтальное масштабирование, как правило, достигается за счёт использования кластеризации и запуска нескольких экземпляров процесса, что требует дополнительной настройки и управления взаимодействием между ними. В Bun данный аспект упрощается за счёт более высокой производительности одного процесса, что позволяет обрабатывать большее количество запросов без необходимости масштабирования на уровне приложения.

Наконец, важную роль играет управление памятью. В сценариях с большим количеством короткоживущих промисов Node.js создает значительное давление на сборщика мусора V8. Bun, благодаря особенностям JavaScriptCore, использует более эффективные механизмы аллокации и повторного использования объектов, что уменьшает частоту срабатывания сборщика мусора и повышает пропускную способность системы. Таким образом, выбор среды выполнения определяется требованиями к системе. Node.js является более универсальным и стабильным решением благодаря зрелой экосистеме, тогда как Bun представляет собой перспективную платформу для высоконагруженных приложений, где критична производительность обработки асинхронных операций.

Список использованных источников:

1. Libuv Documentation : [сайт]. – 2025. – URL: https://docs.libuv.org/en/v1.x/ (дата обращения: 25.03.2026).
2. Bun Runtime Documentation : [сайт]. – 2025. – URL: https://bun.sh/ (дата обращения: 25.03.2026).