Efficiency Evaluation of Node.js Web-Server Frameworks

Danil Demashov[1][0000−0002−4728−7278] and Ilya Gosudarev[2][0000−0003−4236−5991]

[1] ITMO University, Saint Petersburg, Russia, d.danil.s.96@gmail.com
[2] ITMO University, Saint Petersburg, Russia, goss@itmo.ru

Abstract. This article deals with the problem of choice of tools on the JavaScript-based platform Node.js. The platform is aimed at solving a wide range of tasks which include web applications development. That could be done with bare platform modules but to increase development speed and reduce the number of errors an additional layer of abstraction over the network interfaces of the platform is used in the form of small libraries and frameworks. However selecting the right framework often takes a lot of time. That is so because the Node.js ecosystem introduces a great variety of such solutions from independent developers. Thus a huge problem arises: application developers should make a lot of researches on the existing frameworks. This study partly solves that problem in several steps. First, this article presents a classification of frameworks by type for their clustering. Then, the performance of frameworks is obtained but it is not measured directly because such benchmark cannot be examined on the frameworks themselves. Therefore a template application is introduced and implemented on each framework. Consequently all the systems are stress tested. As a result of the study, the measurements and conclusions on them are suggested. Obtained measurements could be applied in a form of a basis for further researches or use recommendations for developers.

Keywords: JavaScript · Node.js · framework · performance · REST · classification

1 Introduction

This article is devoted to the Node.js server frameworks performance study represented by the example of simple web applications which implement REST (Representational State Transfer) API (Application Programming Interface). The main goals are forming frameworks' efficiency rank of and also finding recommendations for these frameworks usage. In order to achieve these aims several steps are required which are: determine framework types; choose application model and define minimal set of modules; select optimal productivity criterion.

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-2590/short14.pdf

REST architecture design template is in high demand for many years. At the present time it makes integral part of single page, mobile applications and micro services server side organization. Particularly it is essential for this kind of applications to be constructed with restricted amount of resources and more importantly without implantation of bulky technologies. However, to increase the development speed and reduce the number of errors, an additional layer of abstraction over the network interfaces of the language is used. These are small libraries and frameworks, which will be discussed later on the Node.js example.

Building a RESTful API can be performed using a variety of languages and platforms for compiling and launching them. And yet Node.js pick is a common way to solve this task due to some features, namely asynchronous requests handling based on an event generation, the so-called event-driven runtime [1]: each request emits an event that enters the processing queue and then Node.js completes the response when it has execution time. This avoids process blocking and excessive memory usage. Furthermore this method differs from the more known concurrency model where operating system threads are involved and a unit of RAM is allocated for each request which can lead to memory overflow.

Therefore Node.js platform is a sufficient tool for developing REST systems but thus a more specialized problem of creating applications arises that is the choice of an appropriate framework. This issue appeared because originally, when the platform just started its evolution, there was no large suitable framework for implementing any complexity applications. As a result, independent developers created their own solutions on a basis of Node built-in modules. Consequently, nowadays the platform ecosystem has a large number of libraries and frameworks for building web applications that overlap in capabilities or have only minor differences. This problem forces developers of final products to constantly conduct their own researches on that topic in order to identify suitable frameworks. The article aims to simplify this kind of search.

2 Framework types

Initially it is required to identify types of frameworks since structural features of the framework affect many parameters of a future application including potential size, complexity and scalability. For further research small libraries or frameworks that have the minimal necessary functionality to form a REST application are preferred.

The following types of frameworks by usage purpose are defined:

1. HTTP server library is a set of functions that grant the ability to build applications based on them; due to the fact that these libraries allow to organize web systems, they belong to the class of frameworks, but they are not completely such [2].
2. API framework is a set of functions which is a skeleton for developing an application that can be accessed via a specific syntax interface.

3. HTTP server framework is an extension of the HTTP library forming an additional layer of abstraction over HTTP commands and also providing a frame of a server organized by such functions [2].
4. MVC framework is a framework that is used to create applications based on the MVC (Model-View-Controller) pattern; this architecture determines how the application will be built.

At the first stage of the study Fastify developers and TechPower researches [3] on determining the frameworks performance were explored. Then repositories containing the code of libraries and frameworks in the Node.js ecosystem were investigated. Table 1 summarizes these analyses and includes GitHub rating, which was obtained at the time of writing, as it represents how often these frameworks are applied in projects. Rates are shown in brackets next to the names of the frameworks and are measured in thousands. The types of frameworks, their possible utilization, advantages and disadvantages are also revealed.

Table 1. Framework classification by types

Type	Purpose	Example	Advantages	Disadvantages
HTTP server library	Apps, Simple REST API	Express (46), Restify (9.5)	Small size, flexibility, quick API creation	Limited functionality
API framework	Apps, API	Loopback (13), Fastify (12.5)	High speed request handling, REST-API generation	Lack of a strict project structure
HTTP server framework	Apps, API	Koa (28), Hapi (12)	Modern request processing, own plugins	Modularity forces to connect a large number of libraries
MVC framework	Apps	Sails (21), Nest (21)	Strict project structure	Snap to MVC approach
Fullstack framework on API framework	SPA	Next (42), Nuxt (23)	Facilitates the backend creation for React or Vue	Inability to select server or client frameworks

On the basis of frameworks documentation and their description, the most suitable frameworks for comparing performance based on REST applications are listed below [4]:

1. Express is the most famous framework of the Node.js platform which was edited and reissued for a plenty of times; on its basis both simple APIs and large applications are built; it is also often the foundation for other frameworks, for example Nest.
2. Restify is a framework for organizing the right RESTful services, the syntax and construction of which is similar to Express by design.

3. Loopback is a large project that has currently moved to the fourth iteration, which has its own engine, automatic API generation and native support for Typescript.
4. Fastify is considered as the fastest framework for developing applications on Node.js.
5. Koa is a framework with an intentionally minimal set of functions, modular form and a new request transfer through middlewares unlike Express.
6. Hapi is a framework with built-in functionality that allows to develop any applications expandable by Hapi's plugins.
7. Sails is a popular MVC framework that imitates a framework like Ruby on Rails but focuses on the informational API and service architecture.
8. Nest is Typescript-based framework that combines object-oriented and functional application structures, is an abstraction over Express or Fastify depending on the choice of the developer; it allows building both simple APIs and large applications.

3 REST API application template

The problem of comparing the quantitative characteristics of frameworks arises because all frameworks differ in built-in functionality, organization of interaction with requests, creation of a requests queue, extension libraries. In order to unify the task of performance measuring, in this study it is proposed to introduce a general application template that will be implemented with different frameworks but which will provide a standard functionality.

REST API applications are in most cases used to provide clients with remote resources. In this study the system is defined which organizes the CRUD (Create, Read, Update, Delete) process for the application log. Also it is necessary to add static files middleware for subsequent performance measurements on requests for the formatted text and files.

The application state management should be taken into account as well. Although the REST architecture implies an absence of storage for client states, session information or a websocket may be in need for storing during data transfer to the client. In this case, the types of these applications are distinguished: with a state where this kind of data is stored in a running application, and without state where such information is stored in special repositories, for example, Redis.

Node.js frameworks have a similar structure, so the application composition can be pre-determined as:

1. Logger is a middleware which saves the information about requests.
2. Bodyparser is a module that selects a request body which is required for Create and Update operations.
3. Logger is a middleware which saves the information about requests.
4. Static handler is a middleware issuing static files from the public directory.
5. Application state storage.

6. Router which processes the requested paths for further the task allocation to functions; in this application should include:
   (a) Path for index.html.
   (b) Paths that form CRUD operations, in other words application APIs.
7. Log storage middleware for the json file.
8. The errors handler which were generated along the request processing, for example, 404 Bad Request or 500 Internal Server Error.

Figure 1 shows the Activity Diagram which explains server actions for making a response to client requests or to a client application.

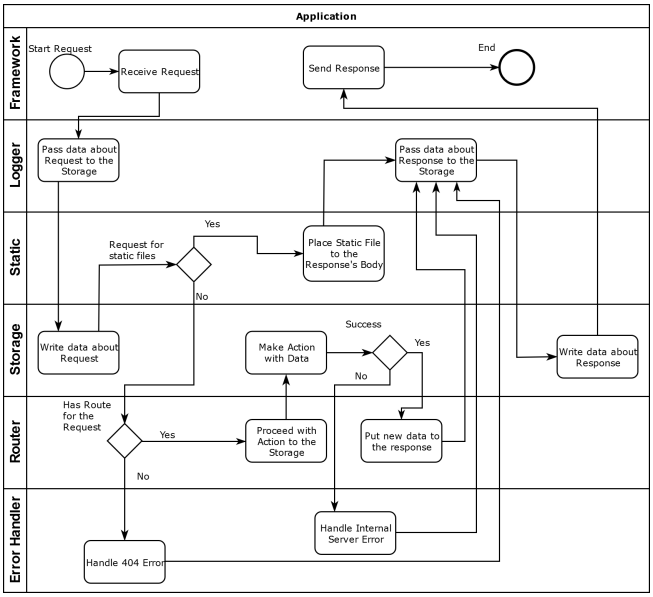

Fig. 1. Activity diagram of the REST API application.

4 Performance measurement plan

Application efficiency is characterized by a large number of its parameters including the technology's effectiveness, the architecture organization accuracy, resistance to external loads and so forth. Therefore it is necessary to adhere to a uniform approach of metering.

There are two approaches in the system performance testing. The first one is simulating a workload by means of artificial scenarios of the application's utilization. The second one is beta testing which is carried by end users. In this work the simulating approach was selected since it allows controlling each round of tests and varying measurement parameters.

Moreover there are several types of performance testing that web applications may be brought to, which are presented below [5]:

1. Load testing is a performance assessment which is done by the exterior system that makes repeated transactions to the application. This method permits to quickly identify application's bottlenecks and present its operating quality.
2. Stress testing is a subtype of the load testing which differs from it in the kind of evaluation: stress testing is carried in the application's conditions that exceed its limits of the normal functioning.
3. Stability testing is a kind of load testing that is conducted in the long period of time in order to detect running stability of the application.

In this paper, the load testing approach was chosen as the most reproducible way of the performance determining for different applications. Stress testing and stability testing to a greater extent show the fault tolerance of web applications but not their quantitative efficiency parameters.

It is important to indicate that Node.js processes originally run on one core [1], however a special platform module can be implemented to use several cores. Due to the fact that all frameworks are launched on a single platform, their behavior on one core could be extrapolated to several cores. Testing was carried out on the equipment, characteristics of which are presented below:

1. CPU: Intel Core i5-4200U CPU @ 1.60GHz 2.30GHz.
2. OS: Ubuntu 18.04 x64.
3. 8 GB RAM.
4. 128 GB SSD ROM.
5. Node 12.13.1.

It should be noted that there are several basic metrics of the application undergoing stress testing, namely: consumed processor time and RAM units, the disk subsystem usage, processed number of requests per time unit, response time. This study is focused on determining characteristics that are irrespective of the physical platform and can be measured both on the application side and on the client. Thus selected application parameters are processed requests per second and the application response time in milliseconds.

Moreover there are universally recognized tools for measuring performance. In this research it is suggested to use the same platform for testing system as for creating applications and utilize performance measurement tools which are available for Node.js. One of the most popular load testing utility on this platform is AutoCannon [6] which imitates client connections and follows some actions scenario for them.

5 Results

Further the created systems were benchmarked in order to detect the most effective framework. Several tests have been conducted simulating clients connections

to the application. The measurements started from 25 connections and ended with 1000 by the increment of 25 connections. In order to combine previously selected load testing metrics for a better representation, this study proposes to divide the number of processed requests by the response time. The results of the measurements are presented in Figure 2.

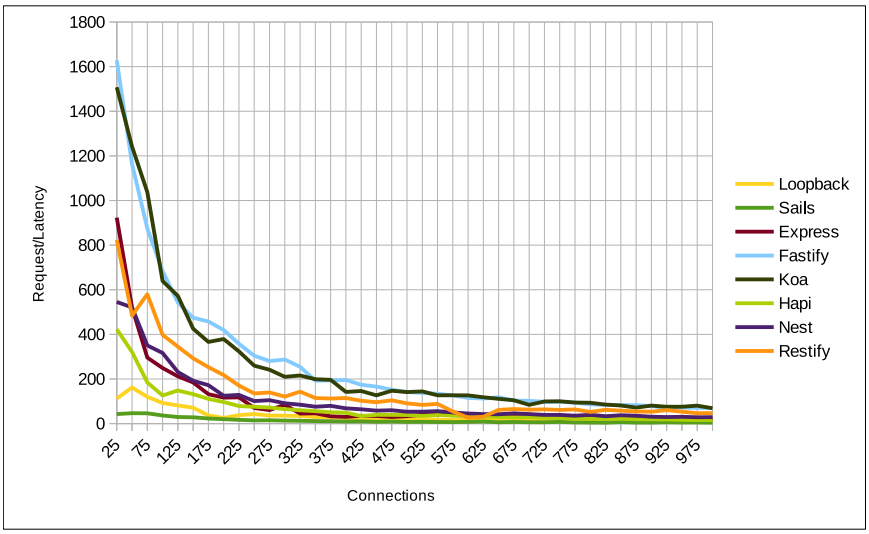

Fig. 2. Overall load testing measurement results.

The graph depicts that the best results are shown by modular frameworks Fastify and Koa, while the slowest ones are large frameworks Loopback and Hapi. The chart also displays that after about 300 connections, a sharp decrease of the frameworks performance becomes more gradual. Thus for this research testing stand 300 connections is the most optimal criterion and therefore Table 2 presents the framework characteristics for this round of tests.

Table 2. Framework characteristics for 300 connections.

Place	Framework	Responses	Latency
1	Fastify	9198	32.03
2	Koa	7890	37.66
3	Restify	6012	49.57
4	Nest	5203	57.13
5	Express	5021	59.27
6	Hapi	4392	67.87
7	Loopback	3257	90.9
8	Sails	1973	150.16

Proceeding from the data which was obtained during the applications creation and their testing, the following frameworks and libraries usage recommendations are formulated:

1. It is proposed to utilize Restify for creating temporary local testing servers which transfer files of single-page applications.
2. To build the API, it is recommended to employ Fastify which provides tools for the fast endpoints dynamic deployment which service clients' demands in calculations which do not depend on the file system and external resources.
3. To organize simple applications that provide static files and have simplified controller structure, Express or Koa are an effective way to create these systems. It should be noted that a faster solution is Koa.
4. For large systems development, the most appropriate choices are Nest, Hapi, and Sails, that have sufficient functionality which is necessary for creating complex applications with non-trivial routes configuration, authorization and client interfaces formation support. Moreover, these frameworks dictate the structure of projects, which helps retaining large projects maintainability. However, it should be noted that the most productive solution is based on Nest.

6 Conclusion

In this article types of frameworks were selected to classify these frameworks in accordance with their purpose. For each type, representatives were identified as well as its features in the form of advantages and disadvantages. Due to the fact that abstract frameworks load testing execution is extremely difficult without an existent application, a template of the REST system was introduced and then implemented on all frameworks. Finally all applications were benchmarked with Node.js tool and use cases were proposed based on the frameworks classification as well as on the carried load testing.

The study can be used as the foundation for further researches or for the initial choice of web project framework.

References

1. Node.js About, https://nodejs.org/en/about. Last accessed 10 Nov 2019.
2. Cantelon, M., Harter, M., Holowaychuk, T. J., Rajlich, N.: Node. js in Action. 2nd edn. Manning, Greenwich (2017).
3. TechPower's web framework benchmarks, https://www.techempower.com/benchmarks/. Last accessed 25 Nov 2019.
4. Loopback frameworks comparison, https://loopback.io/lb3/resources. Last accessed 21 Nov 2019.
5. Matam, S., Jagdeep, J.. Pro Apache JMeter: web application performance testing. Apress, 2017.
6. Benchmarking tool AutoCannon, https://github.com/mcollina/autocannon. Last accessed 30 Nov 2019.