JOIV : Int. J. Inform. Visualization, 7(4) - December 2023 2240-2246

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Node.js Performance Benchmarking and Analysis at Virtualbox, Docker, and Podman Environment Using Node-Bench Method

I Putu Agus Eka Pratama [a,*], I Made Sunia Raharja [a]

[a] *Department of Information Technology, Faculty of Engineering, Udayana University, South Kuta, Badung, 80232, Indonesia*

Corresponding author: [*]eka.pratama@unud.ac.id

***Abstract*—As an asynchronous runtime environment (interpreter) for the development of scalable JavaScript-based network applications, it is necessary to know the performance of the web framework on Node.js in a virtualization-oriented development environment and a container-oriented development environment. This research aims to compare the performance of Node.js in several frameworks in VirtualBox, Docker, and Podman environments. The testing was carried out using some materials like a bench utility at Node Package Manager (NPM) involving the Adonis, Connect, Express, Fastify, Foxify, Hapi, Koa, Molecular, Plumier, Restify, and Sails frameworks, using Object Relational Mapping (ORM) and Raw Query Bookshelf, Knex, MySQL, MySQL2, and Sequelize at Ubuntu Linux operating system. The method research used in this research is the Node-Bench method with requests, latency, and throughput parameters. The testing results show that the best performance score is the Fastify framework with the Sequelize library (ORM) in a container-oriented development environment (Docker and Podman), and the worst performance score is the Express framework with the Mysql2 library (Raw Query) in a virtualization-oriented development environment (VirtualBox). Based on the testing results, developers who use Node.js are more advised to use the Fastify framework with the Sequelize library (ORM) in a container-oriented development environment (Docker or Podman) to obtain better performance. For further research, the implementation and testing at container-oriented development can use cloud-based service (IaaS cloud or PaaS Cloud) for the read-only immutable environment, scalability, and security reasons.**

***Keywords*— Docker; Node-Bench method; Node.js; Podman; VirtualBox.**

Manuscript received 22 Apr. 2023; revised 3 Jun. 2023; accepted 27 Jul. 2023. Date of publication 31 Dec. 2023.

I. INTRODUCTION

Javascript-based application development is helped by Node.js. Node.js is a platform in the form of a runtime environment (interpreter) that functions to run applications developed based on JavaScript, especially for back-end development[1]. As an interpreter, Node.js has several functions to run or execute various JavaScript instructions on the website's back-end environment. Through Node.js, JavaScript can be run on multiple platforms without going through a web browser[2]. Web developers who use Node.js, generally combine it with HTML and CSS to create interactive websites.

Structurally, Node.js is built using the JavaScript engine and has its library, so it does not have to require a webserver (NGINX/Apache). Node.js uses an event-driven and non-blocking I/O model to handle multiple processes simultaneously (multi-thread and multi-tasking). Node.js can also be run in virtualization-oriented development environments (in this case, VirtualBox) and container-oriented development environments (Docker and Podman), each with its specification requirements.

However, developers need to know the performance of each framework on Node.js when running on different development environments. This research will compare the performance of Node.js in the two development environments, specifically in VirtualBox, Docker, and Podman. The testing method used is Node Bench, including raw SQL queries and Object Relational Mapping (ORM).

Tests were carried out using a bench utility at Node Package Manager (NPM) at eleven Node.js frameworks, namely Adonis, Connect, Express, Fastify, Foxify, Hapi, Koa, Molecular, Plumier, Restify, and Sails. For ORM and Raw Query, Bookshelf, Knex, mysql, mysql2, and Sequelize are used.

There are several previous state-of-the-art research studies from this research in various case studies. The first research describes testing five Node.js back-end frameworks (Koa,

Plumier, Express, Nest, Loopback) using the GET and POST methods, with the results showing that Koa has the best performance recommended for developers[3]. However, this research is limited to the Node.js back-end framework. The second research compares JavaScript performance using PHP, Python, and Node.js against REST, with the results showing that Node.js has the fastest response and PHP has the best performance [4]. This research successfully compares the performance of Node.js with two web programming languages but has not compared the performance of the frameworks within them.

The third research is in the form of testing the performance of the Node.js front-end framework (Angular, React, Vue) for the use of the Application Programming Interface (API) on website programming[5]. The results of this research provide good impact and recommendations for front-end developers, but this research only focuses on the Node.js front-end framework.

The fourth research improvised from the third research to test the performance comparison of the Node.js front-end framework (AngularJS, Aurelia, Ember) accompanied by an analysis of the impact on computing resource usage (CPU, internet bandwidth) using twelve measurement metrics: size, version, data binding, dependencies, controllers, scopes, services, directives, templates, routing, structures, and third-party add-ons, with research showing that Aurelia has the best performance, Ember has the best structure, and AngularJS has the best third-party updates [6]. The fifth research is in the form of testing the performance of four Node.js frameworks (AngularJS, Ember, Knockout, Backbone) using the Model View Controller (MVC) architecture with the results of the advantages and disadvantages of each framework according to MVC rules [7].

The sixth research is a performance comparison test of six object-oriented Node.js frameworks (Dojo, ExtJS, JQuery, MooTools, Prototype, YUI2) using a Quality Test with three measurement matrices (Size Metrics, Complexity Metrics, and Maintainability Metrics) to obtain an assessment of quality, validity, and performance, with research results showing that MooTools gets the best results[8]. The seventh research is in the form of testing seven versions of JQuery in terms of performance and validity, with the test results showing that the latest version of JQuery has the best value in terms of validation, performance, and community support [9]. The results of this research become a consideration for developers in choosing the Node.js version and framework. The eighth research discusses the performance and performance of Node.js along with quantitative analysis on online music servers, where the results show that Node.js has high reliability[10].

The ninth research tests and analyzes the performance comparison of REST and Graph-QL implemented in web-based client-server applications using Node.js with the Express framework, with the test results showing that REST has the best performance and Graph-QL has the fastest response related to requests from clients and network bandwidth consumption [11].

The tenth research describes testing the performance and memory consumption of using Node.js Javascript and Golang on the REST API to then be tested and evaluated using Google datasets with the Wilcoxon test type, paired data t-test, and equivalence testing, with the test results showing that Golang has better performance [12]. The eleventh research examines and analyzes several essential factors in Node.js in website development, including event-driven I/0, single-threaded, and asynchronous programming, where the results of these tests can be input for web developers[13]. The twelfth research developed an academic information system based on Rest API using Node.js and PHP to compare its performance, with the results showing that Node.js has better throughput than PHP[14].

The thirteenth research conducted tests on Node. Js-based web applications complement JavaScript on Node.js with Rust and Web Assembly, where test results show that Rust can provide low-level support for non-blocking operations and hardware access to JavaScript and Node.js [15]. The fourteenth research examines several web-based programming languages for full-stack requirements for use with Node.js and then conducts a performance review, with the result that JavaScript is the best language to use with Node.js when compared to other web programming languages[16].

The fifteenth research describes the results of a comparison and evaluation of web-based application development using Node.js and Python Django, with test results showing that the performance of Node.js is better than Django in terms of the number of requests handled, processing time, and transactions[17]. The sixteenth research describes the benefits of using Node.js to implement docker containers on Expert Assist, accompanied by cloud-based Express and MongoDB frameworks, so that the developed system has higher reliability[18].

The seventeenth research conducted a performance comparison test of Codeigniter and Node.js Express on RESTful API, with the test results showing that Node.js Express is more suitable for systems with high user access compared to CodeIgniter[19]. The eighteenth research conducted a comparative test of the performance of Golang and Node.js as web-based back-end applications with MySQL DBMS, with the test results showing that in terms of response time, Node.js and MySQL were the best, while in terms of memory and resource usage CPU computing, Golang and MySQL are the best[20].

The nineteenth research evaluates the performance of Node.js in a JavaScript-based server-side web development environment from the perspective of multithreading, asynchronous I/O, and event-driven programming models, with test results showing that Noje.js has good complexity and support[21]. The twentieth research on the performance of the Node.js framework from a web browser perspective uses SAPUI5 and JQuery based on HTML5, CSS3, and JavaScript libraries, specializing in Single Page Applications (SPA)[22]. The results of this research contribute in the form of an overview of the performance of the JavaScript framework on the client side via a web browser, especially regarding the GET and POST methods related to URL parsing and query string parsing.

Based on this previous research, one of the main problems in Node.js that has not been discussed is the performance comparison of the Node.js framework in virtualization-oriented and container-oriented development environments. This is important considering that developers use one or both

types of development environments, so they need to be compared. For this reason, using the Node Bench method, this research tested the performance of the Node.js framework in a virtualization-oriented and container-oriented development environment. Each Node.js framework will be tested in both development environments, and then each performance will be measured to obtain information regarding which development environment and which framework is the best and recommended for developers.

II. Material and Method

A. Hardware and Software

This research used several hardware and software. For hardware, the Dell Inspiron 15 Notebook is used with Intel i7-6500U (4) @ 3.100GHz, Intel Skylake GT2 [HD Graphics 520] specifications and 16GB of memory. The Linux Ubuntu 22.04 LTS operating system, VirtualBox, Docker, Podman, Node.js, and Node.js Package Manager (NPM) are used for software.

VirtualBox is an open-source licensed software product from Oracle, which uses the concept of virtualization to run a guest operating system along with its libraries and applications on a computer with a host operating system and hypervisor on it [23]. With the availability of VirtualBox, users can perform virtualization from various operating systems on the primary operating system. Examples of virtualization include running the Free BSD operating system through Virtualbox on Linux Ubuntu, computer network virtualization, server virtualization, embedded hypervisor models, and cloud services (IaaS, PaaS, SaaS) [24]. Figure 1. shows the architecture of a Virtualbox with a host operating system and a guest operating system:

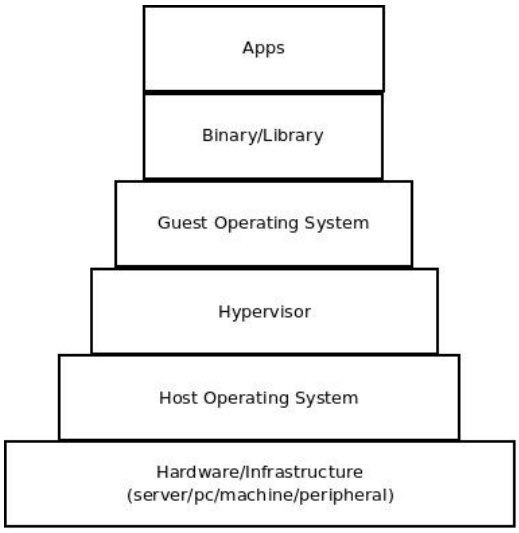

Fig. 1 The architecture of a Virtualbox

Docker is open-source software that facilitates automation in the development and deployment of software into containers, making it possible to virtualize and run applications and operating systems by adding a layer for deployment[25]. Docker consists of 2 parts: the docker server (daemon), which receives requests from the docker client and processes them, and the docker client used by the user.

A Linux container (which allocates computing resources through the kernel) is run by Docker for process virtualization at the operating system level, making it possible to run multiple containers in a single control host and isolated [26]. Figure 2. shows the architecture of Docker.

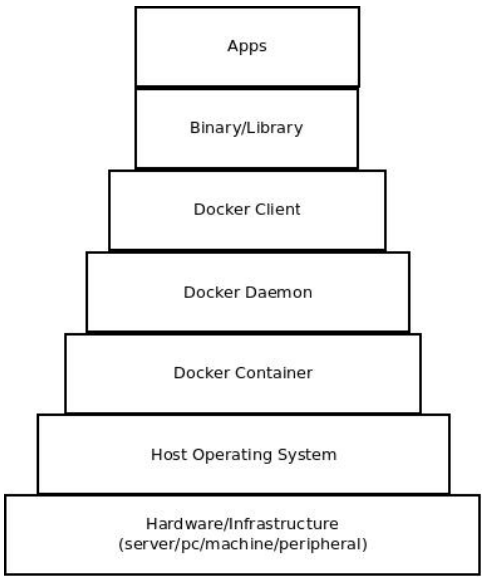

Fig. 2 The architecture of a Docker

Podman is open-source software for managing, discovering, running, and sharing pods, containers, container images, and container volumes based on the libpod library. Podman isolates processes from their surroundings so services can run well and independently. Podman has several commands in the Linux Terminal for easy integration with Docker in system operation and integration[27]. Each Podman has an infrastructure for containers, which aims to connect containers to pods. However, unlike Docker, Podman does not use a daemon for communication. Figure 3. illustrates the structure of Podman:

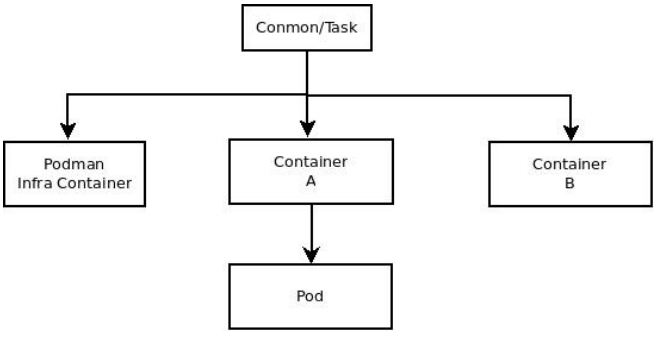

Fig. 3 The architecture of a Podman

Node.js is a runtime environment for JavaScript that is open-source and cross-platform. Meanwhile, Node Package Manager (NPM) is an open-source licensed online package manager and repository for the Node.js framework, which functions to assist developers in package management, model installation, package dependency management, package installation and uninstallation, and running the developed web software [28]. With the availability of NPM, developers can easily and quickly develop web and software

based on Node.js and meet the need for libraries and their dependencies. On Linux operating systems, Node.js and NPM are made available through online repositories on the internet[29]. Figure 4. shows a developer's connectivity chart with Node.js and NPM:

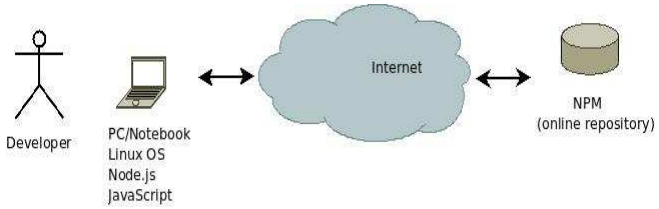

Fig. 4 The connectivity between developer and NPM

B. Node Bench Method

Node-Bench Method is a testing method specifically for website development platforms based on Node.js on a side-by-side basis, which focuses on web performance in terms of frameworks, libraries, and the development environment used by developers [30]. The Node-Bench method is not oriented to network protocols and computer networks, but focuses more on the capabilities of each tested framework, which includes requests, latency, and throughput.

Testing on a web-based software built using Node.js as well as performance testing of a number of Node.js frameworks that use the Node-Bench method, can use open-source licensed libraries and utilities in the form of the Node-Bench library[31] and Nano Bench[32]. which are both JavaScript and Shell Script based.

Testing a Node. js-based software, framework, and library using the Node-Bench method, includes four sequential steps: 1.)Preparing a development environment that supports Node.js and a number of frameworks to be tested, 2.) Installing and configuring the library and utilities supporting the Node-Bench method, 3.)Testing each framework and library according to each case study, 4.)Recording of results, average the results, analysis, and conclusions.

In some cases, testing with the Node-Bench method can collaborate using Object Relational Mapping (ORM) and Raw Query, which are generally supported by several Node.js frameworks. ORM is more widely used in object-oriented case studies to be aligned with relational databases, while Raw Query has functionality and similarities to queries in general but with less memory consumption.

C. Research Flowchart

The research flowchart starts from the preparation stage, where the installation, configuration, and running of the Linux operating system, Node.js, NPM, Virtualbox, Docker, Podman, and the Node-Bench utility are carried out. The Node.js frameworks and libraries that will be tested are run in all three development environments (VirtualBox, Docker, Podman) and then tested based on Node-Bench on requests, latency, and throughput. The test results are stored in three tables, then made into a graphic form, and analyzed to obtain a conclusion. The research flowchart using the Node-Bench method is shown in Figure 5.:

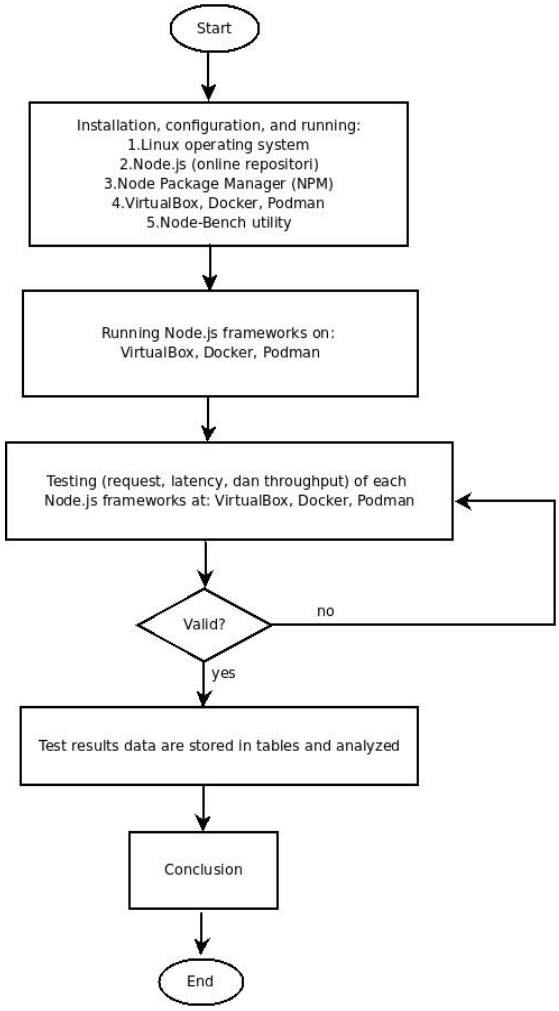

Fig. 5 The research flowchart

III. Result and Discussion

A. Test Result on VirtualBox

The results of testing the performance of Node.js frameworks using the Node-Bench method in VirtualBox for requests, latency, and throughput are shown in Table 1. and the graph from Table I. shown in Figure 6.:

TABLE I
TEST RESULT ON VIRTUALBOX

Framework:Library	Request	Latency	Throughput
adonis:mysql2	7451	13	0.8
adonis:sequelize	4999	19	0.5
connect:mysql2	1400	68	0.3
connect:sequelize	6000	14	1
express:mysql2	789	115	0.2
express:sequelize	2214	44	0.6
fastify:mysql2	1395	67	0.3
fastify:sequelize	7199	13	1.7
foxify:mysql2	1395	67	0.3
foxify:sequelize	6896	14	1.5
hapi:mysql2	938	98	0.2
hapi:sequelize	3275	29	0.5

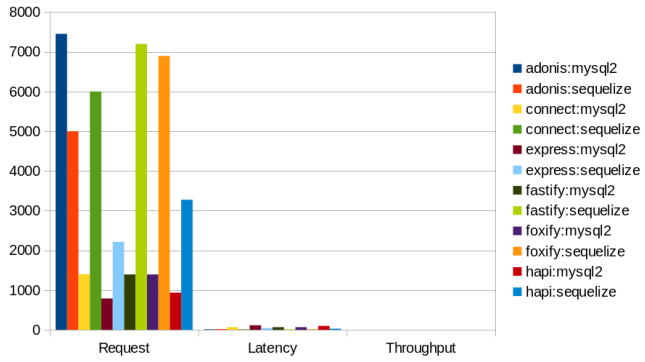

Fig. 6 The graph of test results on Virtualbox

Based on framework performance testing with different libraries in the VirtualBox environment in Table I. and Figure 6., in general, the performance of the Sequelize (ORM) library with the Connect, Express, Fastify, Foxify, and Hapi frameworks is better than the performance of the Mysql2 (Raw Query) library with the same framework. The average performance value of the Sequelize library is almost seven times the performance value of the Mysql2 library. However, the exception is in comparing the Adonis framework for the two libraries, where the performance of the Mysql2 library is almost double Sequelize library.

B. Test Result on Docker

The results of testing the performance of Node.js frameworks using the Node-Bench method in Docker for requests, latency, and throughput are shown in Table II. and the graph from Table II. shown in Figure 7.:

TABLE II
TEST RESULT ON DOCKER

Framework:Library	Request	Latency	Throughput
adonis:mysql2	9650	10	1.0
adonis:sequelize	8496	11	0.9
connect:mysql2	2880	33	0.6
connect:sequelize	10900	8.7	2.5
express:mysql2	1526	59	0.4
express:sequelize	3409	28	1.0
fastify:mysql2	2914	33	0.6
fastify:sequelize	11581	8	2.7
foxify:mysql2	2543	36	0.5
foxify:sequelize	8332	11	1.5
hapi:mysql2	2005	47	0.5
hapi:sequelize	6593	14	1.7

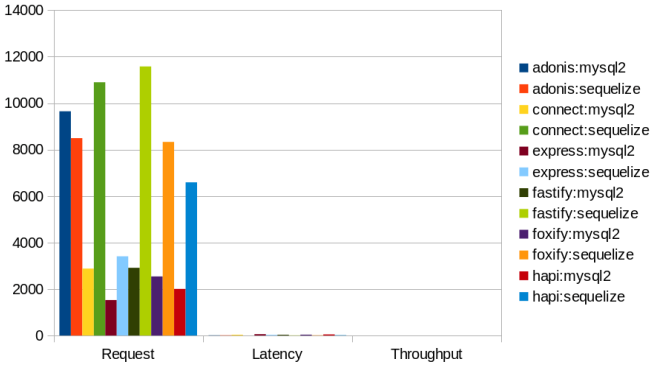

Fig. 7 The graph of test result on Docker

Based on framework performance testing with different libraries in the Docker environment in Table II., in general, the performance for all tested frameworks and libraries is better than the test results on VirtualBox, with an increase of around 10% to 20%. This is because, as a container, Docker only needs to change the configuration and rebuild the container to optimize the running of the framework and libraries.

In addition, the communication at Docker is daemon-based, where daemon connects between developers and containers. In VirtualBox (virtualization), the initial configuration of the guest operating system (which runs on top of the host operating system) is mandatory before configuring the framework and libraries. It causes VirtualBox to consume more computational resources than Docker (container). VirtualBox also does not involve daemons like Docker does.

C. Test Result on Podman

The results of testing the performance of Node.js frameworks using the Node-Bench method in Podman for requests, latency, and throughput are shown in Table III. and the graph from Table III. shown in Figure 8.:

TABLE III
TEST RESULT ON PODMAN

Framework:Library	Request	Latency	Throughput
adonis:mysql2	11137	12	1.2
adonis:sequelize	10218	9	1.1
connect:mysql2	3041	31	0.7
connect:sequelize	9430	10	2.2
express:mysql2	1704	55	0.5
express:sequelize	4799	20	1.4
fastify:mysql2	3161	30	0.7
fastify:sequelize	11496	8	2.7
foxify:mysql2	2914	33	0.6
foxify:sequelize	10591	9	2.3
hapi:mysql2	2009	48	0.5
hapi:sequelize	6463	14	1.7

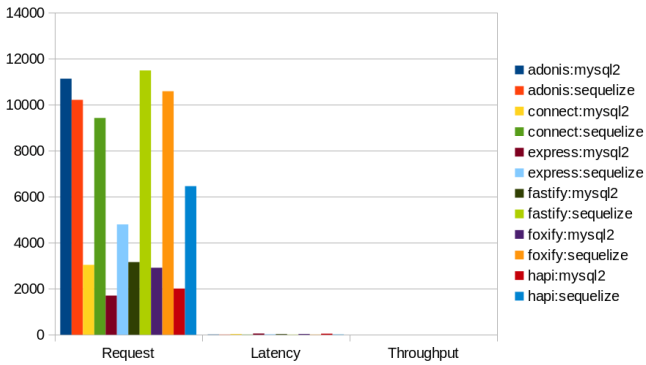

Fig. 8 The graph of test results on Podman

Based on framework performance testing with different libraries in the Podman environment in Table III., in general, the performance for all tested frameworks and libraries is better than the test results on Docker and VirtualBox, with an increase of around 15%. Podman has more value than Docker and VirtualBox, where Podman does not require a daemon for communication like Docker but does not spend computing resources through virtualization and

configuration of guest operating systems as is the case with VirtualBox. Podman only requires process isolation so that it can be more independent from its development environment. By comparing the results of the three tests of Node.js frameworks in the VirtualBox, Docker, and Podman development environments (request, latency, throughput), can be displayed in Figure 9., Figure 10., and Figure 11.:

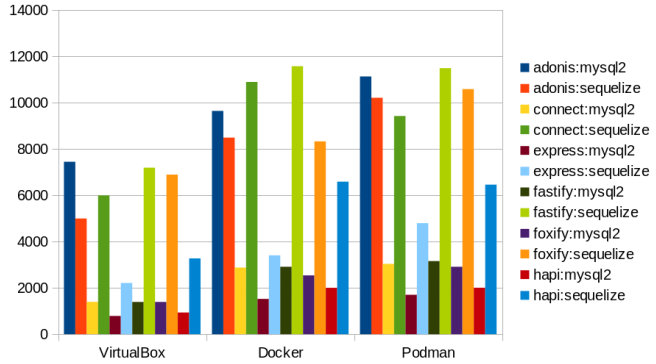

Fig. 9 The performance benchmark (request)

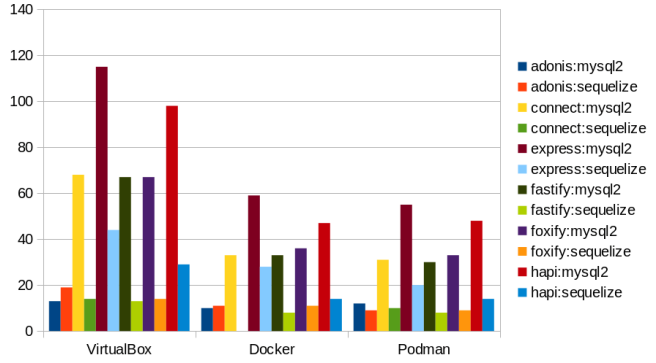

Fig. 10 The performance benchmark (latency)

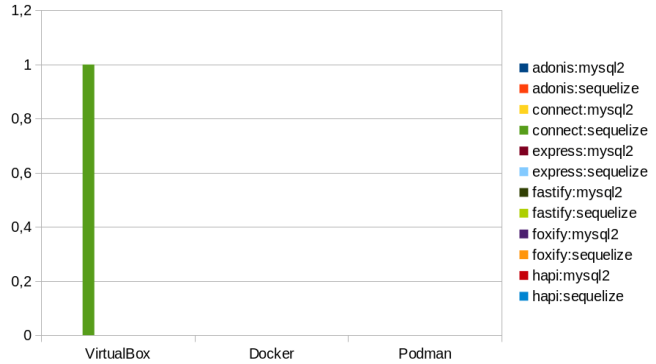

Fig. 11 The performance benchmark (throughput)

The graph in Figure 9., Figure 10., and Figure 11., shows that the performance of the Node.js framework in container-oriented development environment (both Docker and Podman) is better than virtualization-oriented development environment (VirtualBox). This is because the container-oriented development environment does not require guest operating system configurations as is the case with virtualization-oriented development environment, making it more efficient in computing resources and faster.

For the performance of the Node.js framework in a container-oriented development environment, Podman is better than Docker. This is because Podman does not involve daemons like Docker but isolates its processes.

The Fastify framework with the Sequelize library (ORM) in a container-oriented development environment (Docker and Podman) has the best performance score for all frameworks and libraries tested. In contrast, the Express framework with the Mysql2 library (Raw Query) in a virtualization-oriented development environment (VirtualBox) has the worst performance score.

Overall, the contribution of this paper is to provide recommendations to Node. js-based developers in choosing the best framework, library, and Node.js development environment to use in OPM and Raw Query based on the best performance in terms of request, latency, and throughput.

IV. Conclusions

Based on the results, it can be concluded that in terms of the development environment, a container-oriented development environment (Docker and Podman) provides better performance support than a virtualization-oriented development environment (VirtualBox). Meanwhile, the Fastify framework with Sequelize library (ORM) has the best performance in terms of development framework and libraries. Thus, developers who use Node.js are advised to use the Fastify framework with the Sequelize library (ORM) on Docker and Podman to get the best performance that supports smooth development. Developers are discouraged from using a virtualization-oriented development environment such as VirtualBox.

For further research, implementation, and testing of container-oriented development, cloud-based services can be used, especially Infrastructure as a Service (IaaS) Cloud and Platform as a Service (PaaS) Cloud. In this case, the cloud is expected to be able to provide an immutable read-only environment as well as security reasons in both Raw Query and Node.js.

Acknowledgment

Gratitude to the Node.js Community, the Indonesian Linux Community, the FOSS/open source Community, Udayana University, Ananta and the team, and family for the support during this research.

References

[1] D. Herron, *Web Development: Server-Side Web Development Made Easy with Node 14 Using Practical Examples*. Packt Publishing, 2020.

[2] Y. P. D. and W. S. Raharjo, "Performance and Scalability Analysis of Node.js and PHP/Nginx Web Application," *J. Inform.*, vol. 9, no. 2, 2014.

[3] I. P. A. Eka Pratama, "Pengujian Performansi Lima Back-End JavaScript Framework Menggunakan Metode GET dan POST," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 6, pp. 1216 – 1225, 2020.

[4] A. C. Rompis and R. F. Aji, "Perbandingan Performa Kinerja Node.js, PHP, dan Python dalam Aplikasi REST," *CogITo Smart J.*, vol. 4, no. 1, pp. 171–187, 2018.

[5] S. Maganahalli and P. R. R, "Comparison of JavaScript Frontend Frameworks and Web API Services," *Int. Res. J. Eng. Technol.*, vol. 7, no. 6, pp. 108–112, 2020.

[6] J. Ferreira, "A JavaScript Framework Comparison Based on Benchmarking A JavaScript Framework Comparison Based on Benchmarking Software Metrics and Environment Configuration

Software Metrics and Environment Configuration," *Dissertations*, p. 159, 2018.
[7] S. Delcev and D. Draskovic, "Modern JavaScript frameworks: A Survey Study," in *2018 Zooming Innovation in Consumer Technologies Conference, ZINC 2018*, 2018, pp. 106–109.
[8] A. B. Gizas, S. P. Christodoulou, and T. S. Papatheodorou, "Comparative Evaluation of JavaScript Frameworks," in *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web Companion*, 2012, pp. 513–514.
[9] A. Gizas, S. P. Christodoulou, and T. S. Papatheodorou, "Quality and performance assessment of jQuery JavaScript framework," *Proc. IADIS Int. Conf. WWW/Internet 2011, ICWI 2011*, vol. 3, pp. 284–292, 2011.
[10] S. S. Patil and P. S. D. Joshi, "Identification of Performance Improving Factors for Web Application by Performance Testing .," *Int. J. Emerg. Technol. Adv. Eng.*, vol. 2, no. 8, pp. 433–436, 2012.
[11] G. S. Mas Diyasa and G. S. Budiwitjaksono, "Comparative Analysis of Rest and GraphQL Technology on Nodejs-Based Api Development," *Nusant. Sci. Technol. Proc.*, 2021.
[12] H. Ardiansyah and A. Fatwanto, "Comparison of Memory Usage between REST API in Javascript and Golang," *Matrik J. Manajemen, Tek. Inform. dan Rekayasa Komput.*, vol. 22, no. 1, pp. 229–240, 2022.
[13] G. Jadhav and F. Gonsalves, "Role of Node.js in Modern Web Application Development," *Int. Res. J. Eng. Technol.*, vol. 7, no. 6, pp. 6145–6150, 2020.
[14] A. A. Prayogi, M. Niswar, Indrabayu, and M. Rijal, "Design and Implementation of REST API for Academic Information System," in *IOP Conference Series: Materials Science and Engineering*, 2020, vol. 875, no. 1, p. 875.
[15] K. I. D. Kyriakou and N. D. Tselikas, "Complementing JavaScript in High-Performance Node.js and Web Applications with Rust and WebAssembly," *Electron.*, vol. 11, no. 19, 2022.
[16] B. Basumatary and N. Agnihotri, "Benefits and Challenges of Using NodeJS," *Int. J. Innov. Res. Comput. Sci. Technol.*, vol. 10, no. 3, pp. 67–70, 2022.
[17] D. A. Sharma, A. Jain, A. Bahuguna, and D. Dinkar, "Comparison and Evaluation of Web Development Technologies in Node.js and Django," *Int. J. Sci. Res.*, vol. 9, no. 12, pp. 1416–1420, 2019.
[18] S. Sutanto, W. Gunawan, and F. Faeshal, "Arsitektur Container Docker Pada Aplikasi Expert Assist Dengan Teknologi Node.Js, Express Framework & Cloud Database Nosql Mongodb Atlas," *J. Sist. Inf. dan Inform.*, vol. 4, no. 1, pp. 73–89, 2021.
[19] L. Mulana, K. Prihandani, A. Rizal, U. Singaperbanga, and K. Abstract, "Analisis Perbandingan Kinerja Framework Codeigniter Dengan Express.Js Pada Server RESTful Api," *J. Ilm. Wahana Pendidik.*, vol. 8, no. 16, pp. 316–326, 2022.
[20] F. Effendy, Taufik, and B. Adhilaksono, "Performance Comparison of Web Backend and Database: A Case Study of Node.JS, Golang and MySQL, Mongo DB," *Recent Adv. Comput. Sci. Commun.*, vol. 14, no. 6, pp. 1955–1961, 2019.
[21] S. Tilkov and S. Vinoski, "Node.js: Using JavaScript to build high-performance network programs," *IEEE Internet Comput.*, vol. 14, no. 6, pp. 80–83, 2010.
[22] J. Raigoza and R. Thakkar, "Browser Performance of JavaScript Framework, SAPUI5 & jQuery," in *Proceedings - 2016 International Conference on Computational Science and Computational Intelligence, CSCI 2016*, 2017, pp. 1420–1421.
[23] M. K. ANam, D. Sudyana, A. N. Ulfah, and N. Lizarti, "Optimalisasi Penggunaan VirtualBox Sebagai Virtual Computer Laboratory untuk Simulasi Jaringan dan Praktikum pada SMK Taruna Mandiri Pekanbaru," *J-PEMAS-Jurnal Pengabdi. Masy.*, vol. 1, no. 2, pp. 39–44, 2020.
[24] Sutarti, A. P. Pancaro, and F. I. Saputra, "Implementasi IDS (Intrusion Detection System) Pada Sistem Keamanan Jaringan SMAN 1 Cikeusal," *J. PROSISKO*, vol. 5, no. 1, pp. 1–8, 2018.
[25] B. B. Rad, H. J. Bhatti, and M. Ahmadi, "An Introduction to Docker and Analysis of its Performance," *IJCSNS Int. J. Comput. Sci. Netw. Secur.*, vol. 17, no. 3, pp. 228–235, 2017.
[26] B. Gerardus, *Docker Security: A Complete Guide*. 5STARCooks Publisher, 2020.
[27] C. Mukmin, T. Naraloka, and Q. H. Andriyanto, "Analisis Perbandingan Kinerja Layanan Container AS A Service (CAAS) Studi Kasus : Docker dan Podman," *Kumpul. J. Ilmu Komput.*, vol. 08, no. 2, 2021.
[28] Nasution and L. Iswari, "Penerapan React JS Pada Pengembangan FrontEnd Aplikasi Startup Ubaform," *J. Autom. - UII*, vol. 2, no. 2, pp. 193–200, 2021.
[29] D. Guttman, "Fullstack Node.js The Complete Guide to Building Production Apps with Node.js," 2019. [Online]. Available: Fullstack.io.
[30] H. Shah and T. R. Soomro, "Node.js Challenges in Implementation," *Glob. J. Comput. Sci. Technol.*, vol. 17, no. E2, pp. 73–84, 2017.
[31] Isaacs, "Node Bench Library," *Github*, 2022. [Online]. Available: https://github.com/isaacs/node-bench.
[32] Mafintosh, "NPM JS. Nanobench Library," *NPM, 2023*. [Online]. Available: https://www.npmjs.com/package/nanobench.