A Performance Comparison of RESTful Applications Implemented in Spring Boot Java and MS.NET Core

Hardeep Kaur Dhalla*
University of Wisconsin-Stevens Point, WI, USA.
*hdhalla@uwsp.edu

Abstract. RESTful is an emergent architectural style to develop web services/applications due to its lightweight nature. Though more and more organizations are adopting RESTful services, when it comes to choosing the right platform to implement one, teams usually gravitate towards reusing the existing skillset and prefer the platform that they currently support instead of looking for a better performing alternative. Considering that internal architecture and implementation of each platform can drastically affect its efficiency and performance, the objective of this research paper is to assess and compare the performance of RESTful web applications implemented in Spring Boot Java and MS.NET Core. An identical business use case to provide the basic four Create, Read, Update, and Delete (CRUD) operations was developed in both implementations. Apache JMeter 5.2.1, an automated performance testing tool is used in this experiment to create a swamp of concurrent virtual users to load both the applications at regular intervals. The comprehensive experimental results are presented at the end of the research paper.

1. Introduction
Web services are modular software components which are used to develop distributed client-server applications to make communication possible between machines [1]. Nowadays, Ubiquity of web applications over the internet has made it possible to facilitate various kinds of web services provided by educational, governmental, commercial, and multifarious other sectors to billions of people around the globe. Performance of web applications plays an instrumental role in the success of an eBusiness solution these days. A web site running slowly or pages failing to load quickly enough leads to a poor user experience and eventually in loss of customers. It is recommended to include performance testing very early in the software development life cycle to make informed decisions at the architectural level to avoid major design issues later [2].

Representational State Transfer (REST) architectural style is a stateless client-server style. It is a hybrid model of many network architectural styles with additional 4 constraints to connect the components providing a uniform interface. These additional 4 constraints are resource identification; use of representations HTTP GET, POST, PUT, DELETE also known as HTTP verbs to provide a uniform interface while manipulating resources, self-descriptive status codes for response, and hypermedia as the engine of application state (HATEOAS) [3]. RESTful web services are more popular due to their lightweight architecture, and their better performance than the simple object access protocol (SOAP) based web services [1].

The main objective of this research is to benchmark the performance of a RESTful application implemented in two emergent technologies, Spring Boot Java and MS.Net Core, using Apache JMeter automation testing tool. The architecture of this experiment is designed to mimic the production like environment. The obtained results would help technical architects in choosing a high performing,
reliable and scalable technology to implement their RESTful applications. Furthermore, these research results would be beneficial to the instructors and students in the process of learning the implementation of RESTful applications/web services using right technology in the advanced programming courses.

2. Literature Review

In [1], the authors have conducted research to do the performance analysis of RESTful versus SOAP web services using Mercury Load Runner testing tool. Both Java based web services were implemented with apache tomcat server and MySQL database server. It is concluded that RESTful web service performs better than SOAP. Another study [4] has been done to load test different web frameworks Struts 2.0, Spring 3.0 MVC, JSF 1.2x and Wickets using JMeter testing tool. The result of the comparative analysis of performance in this study based on throughput, response time and bandwidth metrics concludes that JSF gives maximum throughput, winner of average response time is Spring, and Wickets consumes most of the bandwidth.

In the study [5], Java performs better than .NET in terms of response time and memory utilization. Load testing was done on the login scenario of both applications using Parasoft WebKing and Hp LoadRunner tools to benchmark the performance of Java EE Struts and VB.NET applications implemented in model view controller (MVC) and 3-tier architecture. In [6][7] authors have done performance testing on Java and ASP.Net applications using Mercury LoadRunner 8.0 testing tool. It is concluded that .Net application is stable and performs faster in response time than its counterpart written in Java technology.

Apache JMeter is used by the authors in the research [8] to do the performance analysis of REST API implemented in MS.NET using entity framework with MS SQL server and Java EE Jersey API using jdbc with MySQL database, but intention of the authors was never to benchmark the performance of both applications. In the obtained results, only HTTP GET operations perform better in Java as compared to other operations in MS.NET. Nevertheless, no research has been conducted in the past to benchmark the performance of RESTful web applications implemented in the technologies Java Spring Boot and MS.NET Core via Create, Read, Update and Delete (CRUD) operations using Apache JMeter performance testing tool. Because it is recommended to run the performance tests in a production like environment [5], this research uses dedicated machines for the database server, web server and JMeter client.

3. Research Methodology

Software testing is one of the critical phases in the software development lifecycle which should be done at regular intervals repeatedly to deliver a reliable and high-quality product to the business users [5]. To evaluate the quality of the software, functional and non-functional requirements provided by the business users should be validated and verified. Although functional requirements are critical, the non-functional requirements influenced by metrics such as scalability, reliability, performance etc. also play a pivotal role in the success of any software. A system performing poorly will not be acceptable to the users in spite of meeting all the functional requirements. The performance testing can be done after system integration to comprehend how an application would behave if subjected to its intended workload or even beyond that. This is usually accomplished by increasing the load on the system to find out the breaking point [9].

Load, stress, and capacity testing can be used to analyze the performance of an application. Load testing is done by subjecting an application to a reasonable workload from zero to intended number of concurrent virtual users. The stress testing is performed to find how stable and robust an application is beyond the intended peak load. It helps in uncovering bugs that make their way into the system under unreasonable workload conditions. Performance parameters like response time, throughput, and utilization of different resources is measured to find bottlenecks, memory leaks, and deviation from the performance objectives. The obtained results are used to fix bugs, optimize, and tune the performance of an application [5].

Plethora of testing tools are available in the market to mimic the behavior of real-world users to analyze different performance parameters before deploying the system in user’s environment. This
research uses Apache JMeter [10] an open-source Java-based performance testing software to simulate the virtual load on the server from normal to heavy operational conditions to analyze the performance of a web application.

Apache JMeter 5.2.1 [10], an automation testing tool, can generate an artificial workload using multi-threading in Java. Any web/application server can be saturated by the load when inundated by the requests made by the virtual users. Multiple report listeners can be exploited in the JMeter test plan to measure a variety of performance metrics such as average response time, throughput, standard deviation, error% and many others. This platform independent tool supports different protocols like HTTP, HTTPS, FTP, SMTP etc. It can run in both gui and non-gui mode, though non-gui is recommended for heavy load testing. Detailed performance analysis reports can be generated in .xml, .csv and .html formats after running the tests.

4. Performance Metrics

Though using JMeter listeners a variety of performance metrics can be monitored for different purposes. Nonetheless, in this research average response time, and error% performance metrics are used as a criterion for the performance evaluation of a RESTful web service. Response time is defined as the total time taken to serve the request made by the client to the web service. It includes the time taken to send the request from client to the server machine and time taken to get the response from the server to the client machine [8]. However, average response time metric is the average of time elapsed to serve all requests made by virtual users in one JMeter test run. It is measured in milliseconds (ms). Error% metric is used to measure the percentage of failed requests in making server connection [10]. Although, throughput metric helps in evaluating availability of an application by observing number of requests served in a unit of time at the server end, but it has a linear relationship with average response time [7]. In the view of this knowledge, throughput is not considered in the analysis of the results. To monitor the health of servers Perfmon JMeter plugin [11] is also used to measure the usage of resources such as CPU, and memory.

5. Experiment Design

The focus of this research is to use Apache JMeter 5.2.1 testing tool to benchmark the performance of the two identical RESTful web applications, which are implemented in MS.NET Core and Spring boot Java technologies, by means of load testing.

![Figure 1. High-level Architecture of the experiment](image)

The MS.Net core RESTful application is implemented using web api (Application Programming Interface) template in the Microsoft Visual studio 2019 and the code is written in C# programming language with targeted framework MS.NET Core 3.1. ADO.NET technology is used to connect to the sql server database. Spring Boot Java 2.1.6 RESTful application is created using Spring tool suite 4.5.1 Integrated Development Environment (IDE) and JDBC technology (mssql-jdbc 6.1.0.jre8) is used for the database connectivity. Four CRUD operations are implemented in both Restful implementations using same use case. Application development servers: Embedded Tomcat in STS Spring source tool suite and IIS express in Visual Studio 2019; are used for testing only during development phase.

In this experiment, Machine 1 is used as a database server; both web applications access the Product table in the AdventureWorks 2014 database on the MS SQL server 2017. SQL script is used to generate 300000 rows with dummy data in the Product table of AdevntureWorks2014. To test the application in the production like environment, both applications are deployed on the external application servers running on the Machine 2; Apache Tomcat 9.0.31 for Java and IIS server 10.0 for
MS.NET Core. For MS.NET core, ASP.NET hosting bundle is installed on the Machine 2 and IIS manager console is used to add a website in the application pool of IIS server. Machine 3 is used to run the JMeter 5.2.1 client in the non-gui mode using command line to do the load testing. Table 1 shows hardware specifications of all the machines used in the experiment.

| Machine      | 64-bit Operating system | Processor        | RAM          | Processor speed |
|--------------|-------------------------|------------------|--------------|-----------------|
| Machine 1    | Windows 10 Home         | Intel core i5-7200U | 16 GB RAM    | 2.50 GHz        |
| Machine 2    | Windows 10 Home         | Intel core i5-3317U | 12 GB RAM    | 1.70 GHz        |
| Machine 3    | Windows 10 Home         | Intel Pentium N3540 | 8 GB RAM     | 2.16 GHz        |

For each implementation, Java and MS.NET Core, 7 JMeter test plans were created in the gui mode for exponentially increasing number of virtual users from 1000 to 64000. In each test plan, HTTP header manager is added to set the content-type to application-json. Each JMeter test plan had 4 HTTP request sampler added to it for GET, POST, PUT, and DELETE methods. HTTP Request Sampler is used in the JMeter test plan to hit the exposed endpoints of Restful application to perform CRUD operations. CSV Data Config element and Summary Report listener were added to each HTTP Request Sampler. Hypothetical test data set is created and stored in comma separated values (CSV) format files and configured in the JMeter using CSV Data Config element. Summary Report listener is used to capture the metrics such as average response time, and error% etc. and results are saved in the output file with .jtl extension for each run. PerfMon Metrics Collector Listener is used to measure the CPU and memory usage on servers. Load tests are run with the increasing number of virtual users 1000, 2000, 4000, 8000, 16000, 32000, 64000 with the ramp-up period of 30s. The results are presented and discussed in next section.

6. Results and Discussion

HTTP GET request was made to retrieve product record using product id from the product table in the database with the exponentially increasing number of virtual users from 1000 to 64000 for both Spring boot Java and MS.NET Core web services. As shown in Figure 2 and Figure 3, both web applications ran smoothly up to 8000 virtual users and started to fail with error afterwards. As the load reached 64000 users, both implementations became inoperable with 75% connection refusal rate. Performance of average response time was found to be better in MS.NET Core in comparison to its Java counterpart.

HTTP POST request was made in order to create a new product record in the database. A total of 11 database fields were populated with each request. In terms of error%, for 8000 virtual users there is connection refusal for 120 users in Java. When subjected to load of 16000 virtual users, requests to make connection for 6174 virtual users is failed in Java versus refusal for 11628 users in MS.Net core. Similarly, for 32000 users, 23081 virtual users failed to make connection to Java application with 72.13% error as compared to 20278 failed requests for users with 63.37 error%. Both applications perform comparably as the load increases to 64000 virtual users giving 75% error. MS.NET Restful application performs better in terms of average response time metric as compared to RESTful application in Java. Results are shown in Figure 4 and Figure 5.
HTTP PUT request was made to send JSON data in the body of the request for 6 fields to update a single product record in the database. As shown in Figure 6 and Figure 7, Java RESTful application performed better initially for intermediate number of virtual users in terms of error% performance metric, but MS.NET core application performed slightly better as the load increased. In case of Java, we observed a spike in the average response time as we approached 8000 users, which can be attributed to the hardware contention, delayed garbage collection or non-availability of other resources, as this was followed by a sudden drop as well. However, MS.NET core performed better than Java in terms of average response time.
HTTP DELETE request was made in order to delete a product record in the database using the product id field. As the load increased, MS.NET Core RESTful application performed better than its Java counterpart in terms of average response time. Both implementations performed comparably in terms of error% for up to 64000 virtual users. Results are shown in Figure 8 and Figure 9. In this experiment, Perfmon plugin [11] was used to monitor the performance of web server and database server in terms of metrics such as CPU%, and memory usage in bytes for all four HTTP operations. This plugin allows to monitor a specific process using process name or process id and the same approach has been used in this experiment for monitoring tomcat server, IIS server and MS SQL server. Perfmon plugin allows results to be exported to a csv file for further statistical analysis, besides generating graphical reports to visualize the results. From the results shown in Figure 10 to Figure 13, as expected, due to the I/O intensive operations carried on database server by both applications, resources in terms of CPU and memory were also consumed more on database server as compared to application server. MS.NET core application has consumed less resources in terms of CPU and memory consistently in comparison to its Java counterpart for all four HTTP operations on both servers.

7. Conclusion

It can be concluded from the results that MS.NET core provides faster response time besides consuming less resources such as CPU and memory as compared to the Java in terms of all HTTP methods. As the load increased to 64000 virtual users, error% started to fall in the range of 75%-85% and the performance of both implementations across all HTTP methods became unacceptable. Overall, both technologies performed comparably in terms of error%. Choosing a right implementation technology is a challenging task for technical architects. These research results would significantly reduce their burden of selecting an optimal technology for their RESTful implementations. In future work, this experiment can be extended to include other frameworks and programming languages such as Golang along with an analysis on the impacts of varying memory allocations as the default memory allocations were used in this paper.

References

[1] A. Bora and T. Bezboruah, “A comparative investigation on implementation of restful versus soap based web services,” International Journal of Database Theory and Application, vol. 8, no. 3, pp. 297–312, 2015.
[2] R. Ramakrishnan, V. Shrawan, and P. Singh, “Setting realistic think times in performance testing: A practitioner’s approach,” in Proceedings of the 10th Innovations in Software Engineering Conference, ser. ISEC ’17. New York, NY, USA: Association for Computing Machinery, 2017, p. 157–164. [Online]. Available: https://doi.org/10.1145/3021460.3021479
[3] R. T. Fielding and R. N. Taylor, “Architectural styles and the design of network-based software architectures,” Ph.D. dissertation, 2000, aAI9980887.
[4] M. Singh and R. Singh, “Load testing of web frameworks,” in 2012 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, 2012, pp. 592–596.
[5] O. Hamed and N. Kafri, “Performance testing for web based application architectures (.net vs. java ee),” in 2009 First International Conference on Networked Digital Technologies, 2009, pp. 218–224.
[6] M. Kalita, S. Khanikar, and T. Bezboruah, “Investigation on performance testing and evaluation of PReWebN: a Java technique for implementing web application,” IET Software, vol. 5, no. 5, pp. 434–444, 2011.

[7] M. Kalita and T. Bezboruah, “Investigation on performance testing and evaluation of PReWebD: a .NET technique for implementing web application,” IET Software, vol. 5, no. 4, pp. 357–365, 2011.

[8] K. Munonye and P. Martinek, “Performance analysis of the Microsoft.NET and Java-based implementation of REST web services,” in 2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY), 2018, pp. 000 191–000 196.

[9] I. Sommerville, Software Engineering, 10th ed. Pearson, 2015.

[10] Apache JMeter. https://jmeter.apache.org/.

[11] PerfMon. https://jmeter-plugins.org/wiki/PerfMon/