Journal of Information Systems and Informatics
Vol. 3, No. 1, March 2022 e-ISSN: 2656-4882 p-ISSN: 2656-5935
http://journal-isi.org/index.php/isi Published By DRPM-UBD

Optimizing JMeter on Performance Testing Using the Bulk Data Method

Nurullah Husufa[1], Ifan Prihandi[2]

[1,2]Information System , Mercu Buana University, Jakarta, Indonesia
Email: [1]nurul_husufa@mercubuana.ac.id, [2]iprihandi@gmail.com

Abstract

Company X is one of the companies engaged in the telecommunications business. To improve customer service, the company developed an Application Programming Interface). The developed API is used to promote product packages to customers. Sending data via API to millions of customers at the same time, can lead to failures due to the inability of the server to process that data. JMeter is one of the tools that can be used to perform performance testing of an API by providing TPS calculation output. Bulk data files are used as input to process large amounts of customer data on JMeter. Proper thread properties settings affect the TPS value, and the study has managed to achieve the TPS value to be achieved with error rate = 0, which means success.

Keywords: Telecommunication Product Package, API, Performance Testing, JMeter, Transaction Per Second

1. INTRODUCTION

Company X is one of the companies engaged in the telecommunications business. To improve customer service, the company developed an Application Programming Interface). The developed API is used to promote product packages to customers. In the API, it will insert what product packages will be promoted and determine which customers will receive the product package promotion. APIs (Application Programming Interfaces) are the way in which libraries of program code, SDKs, and frameworks are available to programmers [1]. APIs allow data or services to be accessed by others.

Performance testing is used to find performance problems that can come from lack of server resources, improper network bandwidth, inadequate database capabilities, failures or weak operating system capabilities, poor WebApp function design and hardware or software issues that lead to reduced client-server performance No. 2] . Performance testing is a process to measure and /

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

or evaluate performance related to aspects of a software system, including: response time, throughput and resource utilizations [3].

JMeter is a desktop-based performance testing tool from Apache, written in the Java programming language, for load testing web pages, web applications and other static or dynamic sources including databases, files, Servlets, Perl scripts, Java Objects, FTP Servers, and others. One of the metrics that can be measured using JMeter is TPS (Transaction Per Second) or throughput. Throughput can be calculated from the target sampler side [4]. Sending data via API to millions of customers at the same time, can lead to failures due to the inability of the server to process that data. This study aims to optimize the setting of properties on JMeter when performing performance testing and the use of files in the application of bulk data methods consisting of different norms for customers to achieve the expected TPS value with error rate = 0.

Reference [5],[6],[7],[8],[9], explains, JMeter is an open source tool used for load testing. The results of comparisons with several other similar tools can be taken into consideration in choosing the tools to be carried out to carry out performance testing. Reference [10], explains, JMeter is suitable for perfomance testing because it is lightweight and easy to install. This is one of the reasons for the selection of JMeter to support this research. Reference [11], explaining how to set up and run JMeter using a GUI. The difference with the current research is the technique of setting thread properties in more detail and their effect on the resulting TPS results. In addition, another difference that can be seen is from how to run JMeter, in this study, JMeter was done by running a script jmeter.sh in a bash editor to anticipate the amount of data that will be processed using the bulk data file method.

2. METHOD

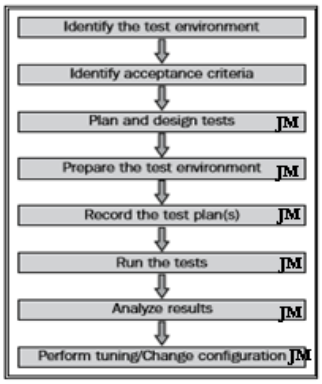

Figure 1. Core performance test activities [13]

Journal of Information Systems and Informatics
Vol. 4, No. 2, June 2022 e-ISSN: 2656-4882 p-ISSN: 2656-5935
http://journal-isi.org/index.php/isi Published By DRPM-UBD

The research will be carried out directly in the performance test section using the JMeter tool. Several scenario tests will be prepared to support this test. The development of the API to be tested by yourself is not included in the research.

In Figure 1, the Performance test will be carried out by doing the following activities, namely

1. Identify the test environment.
   Knowing the test environment such as hardware, software, and network configuration helps obtain an effective test plan and identify testing from scratch.
2. Identify admission criteria.
   Criteria for receiving performance from various modules of the application that are being loaded such as response time, throughput, and resource utilization and constraints. How long does the user have to wait before a particular page is displayed? Response time is something that is considered from the user side, throughput from the business side, and resource utilization from the system side.
3. Test planning and design.
   Know the usage patterns of the application (if any) and design a wide variety of scenarios. Otherwise, consult with stakeholders to understand their business processes and design a possible test plan.
4. Prepare the test environment.
   Configure test environments, tools, and resources to run test plan scenarios. Depending on each company, a separate team may be responsible for managing test tools, and others will configure other aspects such as resource monitoring. In other organizations, only one team is responsible for organizing all aspects.
5. Record the test plan.
   Use the testing tool to record the test plan scenario. The testing tool used in this study is JMeter.
6. Run test.
   Run test plans and verify the correctness of the test scripts and output results. Where test or input data is entered into scripts to simulate the actual data, as well as validate the test data. Pay attention to server logs, through resource monitoring to monitor servers such as warnings and errors. Errors can occur due to problems with test scripts, applications, system resources, or a combination of these.
7. Analysis of results, reports, and retests.
   Check the results of the tests run and identify the cause of the problem. It can come from a system, database, or related application. Related applications can lead to infrastructure changes such as increasing memory, reducing CPU

usage, increasing, or decreasing thread size, database size, and changing network settings. If the cause of the problem is already found, the test(s) should be rerun and compared with the previous results.

3. RESULTS AND DISCUSSION

The research was conducted by testing APIs that require performance testing before the API was released on the production server. The APIs to be tested are:

http://x.x.x.x:xxxx/Genxxx/xx?msisdn=xxx&senderid=yyy

This API will hit a specific customer number to offer a number of product packages that will be promoted. The hit API will generate a log.

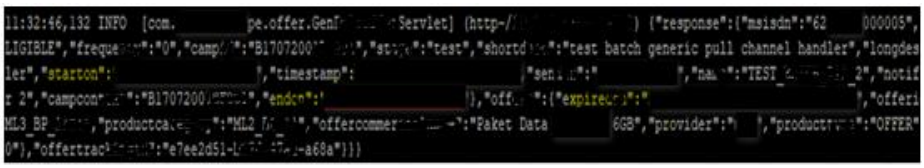

Figure 2. Example logs generated by the API, Genxxx/xx

The detailed activities of the performance test for the Genxx/xx API are as follows:

1. Test environment identification
   Test environments that must be prepared include:
   - The API server is the server where the API will be placed.
   - The JMeter server is the server where JMeter will be placed.
   - JMeter Software.
2. Identify the admission criteria
   The acceptance criteria to be achieved are a throughput of 3000 / tps in a span of 15 minutes and an error rate = 0.
3. Test planning and design
   In the planning and design section of the test, the following things must be prepared:
   - Bulk data files
     Contains the value of each parameter in the API, namely the customer and sender numbers. The number of customer numbers to be hit varies with the amount of 1.5 million data as shown in figure 3.

Journal of Information Systems and Informatics
Vol. 4, No. 2, June 2022 e-ISSN: 2656-4882 p-ISSN: 2656-5935
http://journal-isi.org/index.php/isi Published By DRPM-UBD

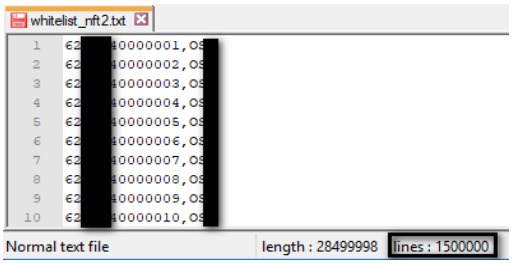

Figure 3. whitelist_nft2.csv Files

- Test scenario to run
  The test scenario to be run is divided into 3 experimental groups. The implementation of the scenario will be seen in the properties change in JMeter. The properties on the JMeter that will be changed include: Number of Threads, Ramp-Up Period and Loop Count.

The test scenario will be saved in a (.jmx) file. The file (.jmx) can be created using the JMeter GUI with the following steps:

1. In **the Test Plan**, right-click, , select the **Add > Threads (Users) menu > Thread Group**

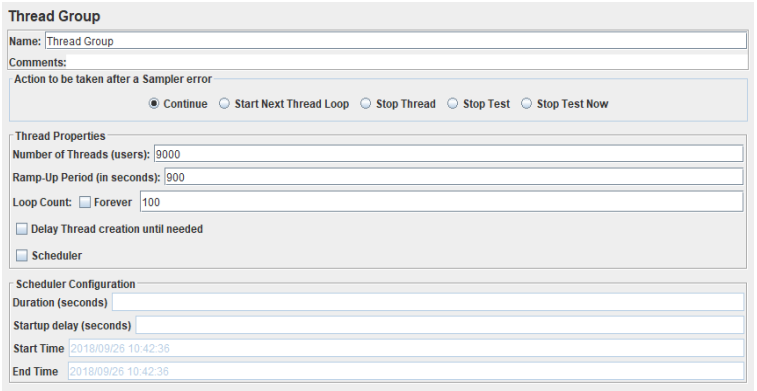

Figure 4. Thread Group Settings

In figure 4, the Number of Threads is the number of users. Ramp-Up Period is a time lag before the next user starts. Loop Count is a loop that will be performed by each user[9], [13]–[15]. To calculate the time lag:

$$\frac{\textbf{Number of Theads}}{\textbf{Ramp-up period}} \quad (1)$$

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

In figure 4., Number of Threads = 9000. Ramp-Up Period = 900, obtained by changing the unit of minutes to seconds, where 15 minutes = 900 seconds and Loop Count = 100. So the throughput is 1000 TPS.

2. In **the Thread Group**, right-click, select the **Add > Sampler menu > HTTP Request**

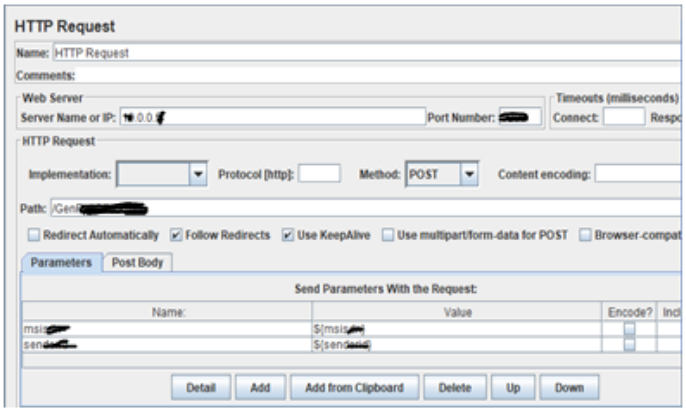

Figure 5. HTTP Request Settings

In Figure 5, IP is the IP of the API. Port number has a default value of =8080. Path is an IP path. Parameters contain the name of the variable in the API that the CSV file will call and the data value from the file written by adding a "${}" sign.

3. In **the Thread Group**, right-click, select the **Add > Config Element menu > CSV Data Set Config**

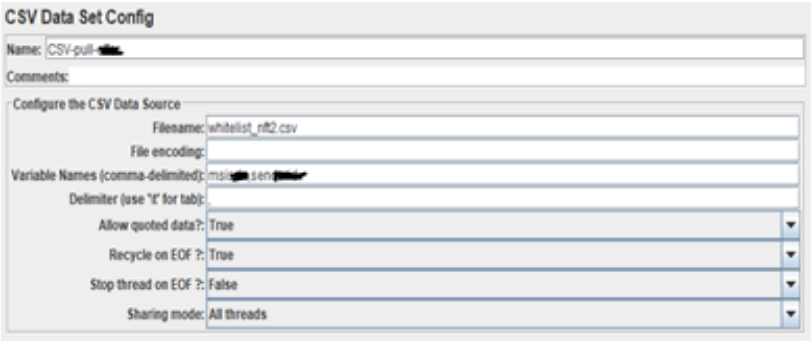

Figure 6. CSV Data Set Config Settings

Journal of Information Systems and Informatics
Vol. 4, No. 2, June 2022 e-ISSN: 2656-4882 p-ISSN: 2656-5935
http://journal-isi.org/index.php/isi Published By DRPM-UBD

In figure 6, Filename is the name of the input data file. Variable Names is the name of the variable in the HTTP Request. Delimiter is a data separator in a file. The CSV file is saved in the /bin folder of Jmeter when running tests. If the JMeter settings have been completed. Select **File > Save**.

1. Prepare the test environment
   Test environtment that must be prepared include:
   - API Server IP : x.x.x.x or x.x.x.x (including user and password)
   - JMeter Server IP : x.x.x.x (including user and password)
     Directory : /x/x/x/apache-jmeter-3.3/bin
   - Bulk data file directory:
     x/x/x/apache-jmeter-3.3/bin/whitelist_nft2.csv
2. Record test plan
   The test plan will be saved on a file with the extension (. JMX) which contains the Number of Threads, Ramp-Up Period and Loop Count of each scenario executed.
3. Run test
   In this study, test plans will be run on the server machine where JMeter is placed. To run JMeter is done by:
   ./jmeter.sh -n -t test_plan_01.jmx -l log.jtl
   Where:
   ./jmeter.sh : default command.
   -n : run jmeter in nongui.
   -t : the file (.jmx) to be run.
   test_plan_01.jmx : the name of the file (.jmx) to be run.
   -l : file for storing log results.
   log.jtl : the name of the log file .

1. Analysis of results, reports, and retests

The results of performance testing carried out on the Genxx/xx API canbe seen in the Table.

Table 1. Test Scenario and Throughput Results Using JMeter – Experiment 1

No.	Test Description	No of Threads	Ramp-Up Period	Loop Count	Throughput Exp.	Throughput Result	Error Rate
1	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_1k_1_1.jtl &	1000	1	1	1000	878.0	0 (0.00%)
2	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_2k_1_1.jtl &	2000	1	1	2000	-	unable to create new native

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

No.	Test Description	No of Threads	Ramp-Up Period	Loop Count	Throughput Exp.	Throughput Result	Error Rate
							thread.
3	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_100_1_30.jtl &	100	1	30	3000	2510.5	0 (0.00%)
4	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_1k_10_30.jtl &	1000	10	30	3000	-	unable to create new native thread.
5	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_9k_300_100.jtl &	9000	300	100	3000	-	unable to create new native thread.
6	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_10k_10_1.jtl &	10,000	10	1	1000	-	unable to create new native thread.

In the Table, JMeter is placed on the same machine as the API, namely the machine with ip x.x.x.x. With the number of threads = 1000, throughput = 878 is achieved with an error rate = 0, but if the number of threads is added to 2000, a warning appears on the jmeter "unable to create new native threads" even though the ramp-up period is the same, which is 1. This warning can be resolved by setting the max user process on the jmeter by writing the following command:

ulimit -u "angka yang akan ditambah"

Table 2. Test Scenario and Throughput Results Using JMeter–Experiment 2

No.	Test Description	No of Threads	Ramp-Up Period	Loop Count	Throughput Exp.	Throughput Result	Error Rate
1	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_30k_10_1.jtl &	30,000	10	1	3000	839.2	7266 (24.22%)
2	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_60k_120_1.jtl &	60,000	120	1	500	248.4	0 (0.00%)
3	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_120k_120_1.jtl &	120,000	120	1	1000	294.7	11225 (9.35%)
4	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_120k_300_1.jtl &	120,000	300	1	400	-	error ='Cannot allocate memory'

Journal of Information Systems and Informatics
Vol. 4, No. 2, June 2022 e-ISSN: 2656-4882 p-ISSN: 2656-5935
http://journal-isi.org/index.php/isi Published By DRPM-UBD

5	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_60k_300_1.jtl &	60,000	300	1	200	124.1	2517 (4.20%)
6	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_36k_3600_100.jtl &	36,000	3600	100	1000	-	error ='Cannot allocate memory'
7	1. Folder script : x@x.x.x.x:/x/x/x/apache-jmeter-3.3/bin 2. List msixxx : /x/x/x/apache-jmeter-3.3/bin /whitelist_nft2.csv 3.Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_9k_900_100.jtl &	9,000	900	100	1000	989.8	473337 (52.59%)

In the Table, JMeter is placed on the same machine as the API, namely the machine with IP x.x.x.x then JMeter is moved on the IP machine x.x.x.x. When Jmeter is still on the x.x.x.x IP machine, the number of threads = 60,000, throughput = 248 is achieved with error rate = 0, but if added the number of threads becomes 120,000 and the ramp-up period is changed to 300, a warning appears on the jmeter "Cannot allocate memory". This warning is resolved by moving the JMeter on the x.x.x.x IP machine and performing optimal scenario settings on the Number of Threads, Ramp-Up Period and Loop Count from the jmeter properties. Too large several Threads and too short or too long a Ramp-Up Period affects the process of implementing performance testing using JMeter, as can be seen in scenario 4 and scenario 6 in experiment 2.

Table 3. Test Scenario and Throughput Results Using JMeter – Experiment 3

No	Test Description	No of Threads	Ramp-Up Period	Loop Count	Throughput Exp.	Throughput Result	Error Rate	Notes
-	Folder script : x@x.x.x.x:/x/x/x/apache-jmeter-3.3/bin							
-	List Msisdn : /x/x/x/apache-jmeter-3.3/bin /whitelist_nft2.csv							
-	API : http://x.x.x.x:xxxx/Genxxx/xx?msixxx=xxx&senxxx=yyy							
1	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_30k_300_10.jtl &	30,000	300	10	1000	840.5	0 (0.00%)	
2	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl2_90k_900_10.jtl &	90,000	900	10	1000	-	error ='Cannot allocate memory'	stop in 75.079 - 00:01:24 - 889,3

3	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_180k_900_10.jtl &	180,000	900	10	2000	-	error ='Cannot allocate memory'	stop in 226.983 - 00:02:26 - 1555,3
4	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_9k_900_100.jtl &	9,000	900	100	1000	988.4	0 (0.00%)	
5	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_18k_900_100.jtl &	18,000	900	100	2000	1939.1	0 (0.00%)	
6	Run script : nohup ./jmeter.sh -n -t Pull-x2.jmx -l hsl_27k_900_100.jtl &	27,000	900	100	3000	2836.5	0 (0.00%)	

In the Table, the throughput result obtained in scenario 6 is 2836.5 close to the desired 3000 number with error rate = 0, which means all data is successful in the process without any failures.

4. CONCLUSION

Based on the results of research on Optimizing the Use of JMeter in Performance Testing Using the Bulk Data Method, Performance testing using JMeter tools helps to find out the TPS (Transaction Per Second) or Throughput of the API (Application Programming Interface) Telecommunications product package developed. The Number of Threads, Ramp-Up Period and Loop Count settings of the jmeter properties affect the Throughput value and the error rate to be generated. The use of CSV files as a method of bulk data containing different customer numbers is helpful for processing large amounts of data simultaneously. The Number of Threads, Ramp-Up Period and Loop Count of the proper jmeter properties will result in the throughput as expected and the number of error rates = 0 thus reducing data delivery failures at the same time.

REFERENCES

[1] L. Murphy, T. Alliyu, A. Macvean, M. B. Kery, and B. A. Myers, "Preliminary Analysis of REST API Style Guidelines," PLATEAU'17 Work. Eval. Usability Program. Lang. Tools, pp. 1–9, 2017.

[2] B. R. M. Roger S. Pressman, Software Engineering: a practitioner's approach. 2015.

[3] Z. M. Jiang and A. E. Hassan, "A Survey on Load Testing of Large-Scale Software Systems," IEEE Trans. Softw. Eng., vol. 41, no. 11, pp. 1091–1118, 2015.

[4] and W. M. O. Sadiq, Muhammad, Muhammad Shahid Iqbal, Amizah Malip, "A Survey of Most Common Referred Automated Performance Testing Tools," ARPN J. Sci. Technol., vol. 5, no. 11, pp. 525–536, 2015.

[5] R. Akiladevi, P. Vidhupriya, and V. Sudha, "A Study and Analysis on Software Testing Tools," vol. 118, no. 18, pp. 1783–1799, 2018.

[6] D. Kelkar and K. Kandalgaonkar, "Analysis and Comparison of Performance Testing Tools," Int. J. Adv. Res. Comput. Eng. Technol., vol. 4, no. 5, pp. 1880–1883, 2015.

[7] M. Sharma, R. Kumar, and V. Mansotra, "Proposed Stemming Algorithm for Hindi Information Retrieval," Int. J. Innov. Res. Comput. Commun. Eng. (An ISO Certif. Organ., vol. 3297, no. 6, pp. 11449–11455, 2016.

[8] T. Suominen, "Performance Testing Rest Apis," 2017.

[9] A. K. Tiwari, S. Yadav, M. Mathuria, M. Sharma, H. Chaudhary, and M. Tech Scholar, "Performance Optimization of Web Applications using Connection Pooling," 2016.

[10] D. M Niranjanamurthy , Kumar S, Kiran , Saha, Anupama , Chahar, "IJARCCE Wireless Sensor Network (WSN): Applications in Oil & Gas and Agriculture Industries in Nigeria," Int. J. Adv. Res. Comput. Commun. Eng. ISO, vol. 3297, 2007.

[11] S. A. Kamarudin, Kusrini, "Uji kinerja sistem web service pembayaran mahasiswa menggunakan apache jmeter (studi kasus: Universitas amikom yogyakarta)," Teknol. Inf., vol. XIII, no. 1, pp. 44–52, 2018.

[12] M. Masse, REST API Design Rule Book, vol. 53, no. 9. 2013.

[13] B. Erinle, Performance Testing with JMeter 2.9. 2013.

[14] K. Gupta and M. Mathuria, "Improving performance of web application approaches using connection pooling," Proc. Int. Conf. Electron. Commun. Aerosp. Technol. ICECA 2017, vol. 2017-Janua, pp. 355–358, 2017.

[15] M. Jeskanen and J. Moilanen, "Non-functional testing : security and performance testing," no. November, 2015.