International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol. 9, Issue.6, June 2021 E-ISSN: 2347-2693

Test Automation Framework for Content Delivery Network

Jayesh Kumar Yadav[1*], Nagaraja G.S.[2]

[1,2]Computer Science and Engineering Department, R V College of Engineering, Bengaluru, India

*Corresponding Author: jayeshkumary.cs17@rvce.edu.in, Tel.: +91-8921083022

DOI: https://doi.org/10.26438/ijcse/v9i6.4548 | Available online at: www.ijcseonline.org

Received: 16/Jun/2021, Accepted: 20/Jun/2021, Published: 30/Jun/2021

Abstract— Content Delivery Network, or a CDN, is a globally distributed network of servers that helps provide good availability, faster and reliable performance, and security to the content distributors. In order to maintain a secure and reliable system any change proposed for the network needs to be thoroughly tested. But testing a full software release takes a lot of time because of a very huge database and complex dependencies between them. This increases the total time taken in the software development life cycle. In the existing system, a tester has to write test scripts for every Change Request (CR), which is a documented request to modify the current software system. This effort can be substantially reduced by developing a tool which can accurately test all the changes by dynamically generating test values for each metadata tag. (In this paper, the term "metadata tag" refers to settings used to control the configuration of web servers). This reduces the time to figure out all complex dependencies and test for each and every change made. The aim is to provide a simple, clean interface which allows the user to select a Change Request he wants to test and then dynamically generate positive and negative test values on which test will run on and provide a detailed result to the user whether the test passed or not.

Keywords**—**Content Delivery Network; Metadata Tag; Change Request

I. Introduction

Testing each and every feature takes a lot of manual effort and hence increases the time taken in the software development lifecycle. Also, figuring out all dependencies of each type of tag and manually writing test cases for it is a tedious task. It also has a higher chance of human error. This makes the whole system inefficient and error prone. Hence there was a requirement for an automated testing tool which can accurately test all the changes by dynamically generating test values for each tag. This tool can solve the problem of human error as the whole system will be automated. This will increase the efficiency, and as a result the whole time taken in the software development life cycle will be reduced. This project is aimed to test the web-server software of Content Delivery Network (CDN). The next section will give a brief introduction about CDN and its working.

1.1 CONTENT DELIVERY NETWORK (CDN) OVERVIEW

Content Delivery Network, or a CDN, is a globally distributed network of servers that helps provide good availability, faster and reliable performance, and security to the content distributors. In a CDN, replica of content is placed near to the user in order to reduce latency, increase scalability and availability of the content providing a streamlined experience to the end user. It helps content distributors to deliver the content to the end-users on behalf of the origin server. With the help of request redirection algorithm, the best replica server is selected and users requests are directed to that server. A CDN also enhances the performance of the web during high traffic by distributing the traffic over different servers in the network. It is also used widely for delivering streaming services such as video on demand and live streaming economically and reliably.

1.2 CDN WORKFLOW

This section describes the workflow of content delivery network. Figure 1 describes the flow of the whole process of distributing content through content delivery network.

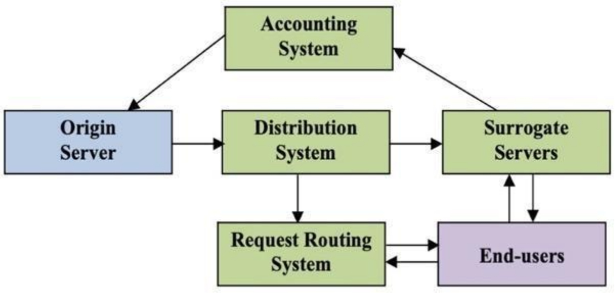

Figure 1: CDN Workflow

Origin server provides the content to the Distribution System for replication. Content is then replicated to surrogate servers by Distribution System and it also maintains the consistency of data at the surrogate servers. The Distribution system also provides the information about replication to the request routing system to help in surrogate server selection for redirecting end users requests. The Request Routing System handles the request for the content from the end user. Then the request is redirected to the suitable surrogate server by the Request Routing System. Then the end-user request is satisfied by the selected surrogate on the behalf of the origin server. Log of data transferred is sent to the Accounting System by Surrogate server. This information is aggregated by the Accounting System for use by the origin server and for billing the content provider. Origin server uses this aggregated information for decision about which contents should be replicated and where in order to further improve web performance as well to minimize the cost.

II. Related Work

Reference [1] talks about Cloud based Content Delivery Network, its architecture and working. It also discusses challenges of design for CDN along with the evolved architecture.

Reference [2] talks about how a content delivery network is used to improve performance and how it helps in improving the challenges of the web. It also talks about pointing out the components, existing emerging paradigms and review of literature on the existing strategies for content distribution.

Reference [3] explains how to improve the efficiency of the content distribution and optimize the overall performance by making full use of the network characteristics and the effective information provided by the network operators.

Reference [4] highlights the role of location, the growing complexity of the CDN ecosystem, and their relationship to, and implications for interconnection markets.

Reference [5] gives a comparative study of the content delivery network and named data networking and the advantages of named data networking over content delivery networks.

Reference [6] explains how to reduce load on the origin server and the traffic on the Internet, and hence improve response time to users. It also talks about optimal placement of CDN servers.

III. Methodology

The step-by-step procedure proposed to develop the tool is as follows.

3.1 Getting the list of Tags depending on the chosen version and the Change Request number

Whenever a code change is done by someone, a corresponding Change request (CR) is generated. This Change Request is then forwarded for testing phase. The list of the CRs with the description has to be populated and the user can select one of the CRs from the dropdown. The next step is getting the list of all metadata tags in a particular CR. (A metadata tag is basically a configuration setting, and determines how a particular request or response will be handled). Then we find out which tags are being changed and populate them and the user can select the tags.

3.2 Getting the data-type and scope of the chosen tag

Different tags accept different data-types as per their functionality. Hence, to generate positive and negative test cases it is necessary to get the type of data that these tags accept. Next step is finding the scope of the tag and deciding which channels are mapped to this tag . A channel is a path through which the particular tag can be pushed/applied to the web-server. So we need to find what all channels this tag is part of.

3.3 Generating combinations to run for each tag and add custom cases

Depending on the type of value the particular tag accepts, various combinations are dynamically generated. For a tag that accepts integer, float or any numerical value, there is no exact defined range as it varies from one tag to another. Hence an option to enter range is given to the user. For example, if a tag is of type "flag", then the accepted values could be "on" and "off", and negative test cases could be an integer or float value. All the test combinations are populated in tabular form where users can also add positive or negative custom tests of their own.

3.4 Generate a test script for all the test cases

A test script is generated automatically which will be used to send all the requests to the test server and record the responses and the requests in log files. Generation of test scripts depends on the channel on which we are testing on. All the individual test cases for each channel having tag and value pair are generated and stored in separate files.

3.5 Running test on user's machine and parsing the result from log file

The test file generated in the previous step is copied to the user's machine. Then the tests are executed on the user's machine. The detailed information about each and every step gets stored in the log file corresponding to that file. These log files have all the information about the request and the response, like size, header, body, status, caching, etc. These parameters are analysed and it is determined whether changes were accepted or not. Then, all the results in a human readable form are displayed to the user.

Figure 2 is the flow chart of the whole process.

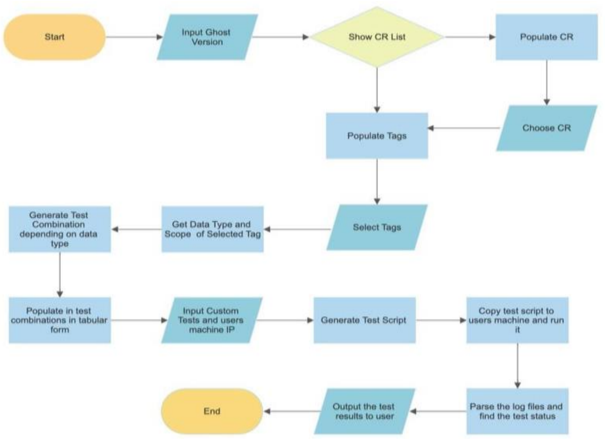
Start
Input Ghost Version
Show CR List
Populate CR
Choose CR
Populate Tags
Select Tags
Get Data Type and Scope of Selected Tag
Generate Test Combination depending on data type
Populate in test combinations in tabular form
Input Custom Tests and users machine IP
Generate Test Script
Copy test script to users machine and run it
Parse the log files and find the test status
Output the test results to user
End

Figure 2:Flow Chart

IV. OBSERVATIONS AND RESULTS

The tool has to test every tag on positive and negative test cases based on what data-type values are accepted by it. The final results should match with the expected results completely for a successful test run. Any mismatch between the expected and the final results means that there is a bug in the Change Request and that Change Request is forwarded for review. Hence, in this way it detects errors in earlier phases of testing.

Below is an example result for four tags accepting values of type double, integer, flag and string respectively. For a test to pass, the test value should be of the same type as the type accepted by the particular tag; otherwise it should fail. In the table, we can observe that the expected result is the same as the final result. Hence the test is marked as 'PASSED' and no anomaly found in the working of the tag.

Table 1. Test Results

Tag	Type accepted	Test value	Expected Result	Final result
Tag1	double	10.45	Pass	Pass
Tag1	double	Test1	Fail	Fail
Tag1	double	12/4/20	Fail	Fail
Tag2	Integer	64	Pass	Pass
Tag2	Integer	57.873	Fail	Fail
Tag2	Integer	Test1	Fail	Fail
Tag2	Integer	24%4	Fail	Fail
Tag3	Flag	on	Pass	Pass
Tag3	Flag	off	Pass	Pass
Tag3	Flag	100	Fail	Fail
Tag3	Flag	15.89	Fail	Fail
Tag4	String	Server-up	Pass	Pass
Tag4	String	245	Fail	Fail
Tag4	String	3.145	Fail	Fail

V. CONCLUSION AND FUTURE SCOPE

This tool automates the process of testing a Change Request, thereby saving time. It makes the whole process more fluid, which can be tedious if done manually. It helps in increasing the efficiency of the testing phase by reducing the chances of human errors. In the final results of this tool, I found out that it was able to test every feature with required accuracy and was able to detect any anomaly in the feature if present in the initial phase of testing. It was observed that time taken to test around 10 test cases was 151 seconds and time taken to test 20 test cases was 280 seconds which is very less compared to time taken if done manually and hence total time taken in software development life cycle. Some of the improvements that can be made to the existing system are Providing user interface to debug in case of test fails, adding option to send the full test result to users email and expanding tool to include more complex white box testing.

ACKNOWLEDGMENT

I would like to express our deep gratitude towards my Guide Dr. Nagaraja G. S and Prof. & Head Dr. Ramakanth Kumar P, for providing their guidance, enthusiastic encouragement, valuable and constructive suggestions during planning and development of this case study paper. Also we whole-heartedly thank our beloved Principal Dr. K. N. Subramanya for their continuous support to carry out this project.

REFERENCES

[1] Q. Jia, R. Xie, T. Huang, J. Liu, and Y. Liu, "The Collaboration for Content Delivery and Network Infrastructures: A Survey," IEEE Access, vol. 5, pp. 18088-18106, 2017.
[2] K. Hosanagar, R. Krishnan, M. Smith and J. Chuang, "Optimal pricing of content delivery network (CDN) services," 37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the, 2004, pp. 10 pp
[3] Pathan M, Buyya R. "A taxonomy of CDNs: In Content delivery networks," 2008 (pp. 33-77), Springer, Berlin, Heidelberg.
[4] Stocker, Volker, Georgios Smaragdakis, William Lehr, and Steven Bauer. "The growing complexity of content delivery networks: Challenges and implications for the Internet ecosystem." Telecommunications Policy 41, no. 10 (2017): 1003-1016.
[5] G. Ma, Z. Chen, J. Cao, Z. Guo, Y. Jiang and X. Guo, "A tentative comparison on CDN and NDN," 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), San Diego, CA, 2014, pp. 2893- 2898.
[6] Pallis, George, and Athena Vakali. "Insight and perspectives for content delivery networks." Communications of the ACM49, no. 1 (2006): 101- 106.
[7] Garmehi, Mehran, Morteza Analoui, Mukaddim Pathan, and Rajkumar Buyya. "An economic replica placement mechanism for streaming content distribution in Hybrid CDN-P2P networks." Computer Communications 52 (2014): 60-70.
[8] Buyya, Rajkumar, Al-Mukaddim Khan Pathan, James Broberg, and Zahir Tari. "A case for peering of content delivery networks." arXiv preprint cs/0609027 (2006).
[9] Wang, Limin, Vivek Pai, and Larry Peterson. "The effectiveness of request redirection on CDN robustness." ACM SIGOPS Operating Systems Review 36, no. SI (2002): 345-360.
[10] Hu, Han, Yonggang Wen, Tat-Seng Chua, Zhi Wang, Jian Huang, Wenwu Zhu, and Di Wu. "Community based effective social video contents placement in cloud centric CDN network." In Multimedia and Expo (ICME), 2014 IEEE International Conference on, pp. 1-6. IEEE, 2014.
[11] Raciborski, Nathan F., and Bradley B. Harvell. "Write-cost optimization of CDN storage architecture." U.S. Patent 8,321,521 issued November 27, 2012.

AUTHORS PROFILE

Jayesh Kumar Yadav is an undergraduate student studying Computer Science of Engineering in R.V. College of Engineering, Bangalore, India. His areas of interest include Web Development and Cloud Computing

Nagaraja G. S is Professor and Associate Dean of Dept. of Computer Science of Engineering in R.V. College of Engineering, Bangalore, India. His areas of interest include computer Networks & Management, Multimedia Communications, Computer Architecture, Protocol Design