International Journal of Engineering

Journal Homepage: www.ije.ir

Investigation on Reliability Estimation of Loosely Coupled Software as a Service Execution Using Clustered and Non-Clustered Web Server

A. Bora*, T. Bezboruah

Department of Electronics & Communication Technology, Gauhati University, Guwahati, India

PAPER INFO

Paper history:
Received 20 May 2019
Received in revised form 01 October 2019
Accepted 08 November 2019

Keywords:
Clustered Web Server
Software as a Service
Multi Tenant Environment
Reliability

ABSTRACT

Evaluating the reliability of loosely coupled Software as a Service through the paradigm of a cluster-based and non-cluster-based web server is considered to be an important attribute for the service delivery and execution. We proposed a novel method for measuring the reliability of Software as a Service execution through load testing. The fault count of the model against the stresses of users is deployed. A prototype application using Simple Object Access Protocol-based web service is developed and the study is carried out over there. The experimental setup, architecture, load testing results followed by a comparative study is discussed in this work. It is observed that the reliability of the service by using clustered and non-clustered web server degrades after a specific limit of stress level execution point. The comparative assessment predicts that the reliability of service by using a cluster-based web server is better than the service with a non-cluster based web server. With an increase in the stress level of usage in a multi-tenant environment, the service with clustered web server delivers better reliability than the service without a clustered web server. The occurrences of HyperText Transfer Protocol request failure in the service with a clustered web server is comparatively less than its counterpart service without a clustered web server. The study helps in identifying the applicability of the method and shows the effectiveness of such deployment.

***doi**: 10.5829/ije.2020.33.01a.09*

1. INTRODUCTION

The Software as a Service (SaaS) is a deployment way for delivering software service to end-users over the network. Instead of following the complex strategy of developing, installing and configuring software modules, the users of SaaS can access the business logic (BL) over the internet [1]. The service providers of SaaS provide the software modules with access rights to their consumers [2, 3]. The users of SaaS can be categorized into multi or single-tenant isolated consumers. Although SaaS can be accessed from different isolated nodes, the demand is comparatively observed in multi-tenant services. The users in multi-tenant environment request simultaneously the instances of SaaS application that is hosted in the shared server. In a multi-tenant environment, the end-users can save time and other resources that are primarily required for maintaining stand-alone services [4–7].

The services of the SaaS platform is provided to users of SaaS application along with authentication and authorization control, secured communication and data storage. The scalability of the SaaS application can be achieved through clustered working nodes having similar instances of the service. The reliability estimation of SaaS execution is an important issue in the research community. In this work, a prototype of SaaS application with segregation of service layers is deployed. The reliability estimation framework, recorded HyperText Transfer Protocol (HTTP) fault count and failure rate for the deployment of the application by using clustered and non clustered web server is discussed. The reliability during high stress of usage is also evaluated in this study.

*Corresponding Author Email: *abhijit.bora0099@gmail.com* (A. Bora)

Please cite this article as: A. Bora, T. Bezboruah, Investigation on Reliability Estimation of Loosely Coupled Software as a Service Execution Using Clustered and Non-Clustered Web Server, International Journal of Engineering (IJE), IJE TRANSACTIONS A: Basics Vol. 33, No. 1, (January 2020) 75-81

2. RELATED WORK

In the research community, the way of evaluating the reliability of SaaS in a multi-tenant environment is still fewer. In the year 2006, Chong had discussed that SaaS application can be classified into different levels based on scalability, the efficiency of the multi-tenant environment and configurability [8]. In the same year, Saddik had developed a performance evaluation framework by using a paradigm of a service-oriented system [9]. The study revealed that quality of service is an important perspective for deploying in the public domain. In the year 2008, Mietzner et al. had discussed the delivery model of SaaS application from the perspective of the business domain and industrial services [10]. In the year 2010, Chen presented a quality evaluation framework for the deployment of SaaS applications [11]. However, he did not present the reliability aspects of such deployment. In the same year, Kang had discussed the importance of service integration and virtualization for evaluating the SaaS model [12]. However, the experimental arrangement and evaluation strategy were not done in his study. In the year 2014, Medhi et al. had highlighted the quality aspects of service delivery in the paradigm of SaaS architecture [13]. The statistical evaluation of the data had established the viability of such an estimation. However, the importance of reliability using clustered web servers was not discussed. In the same year, Candeia et al. had discussed the importance of capacity planning for the deployment of SaaS applications. They raised that long term and short term capacity planning of the deployment machines can lead to profit-making for different business modules. They had highlighted that the acceptable performance of the SaaS application can be achieved through proper capacity planning of system design [14]. In the same year, Rahmani et al. had carried out a study for developing a reliability estimation framework that can be followed for interoperable system modules [15]. In the year 2017, Sharif et al. had presented a strategy for the management of workflow scheduling in a multi-tenant environment [16]. The study had revealed that better cost and time complexity of SaaS execution can be achieved while satisfying system constraints and privacy policy of service delivery. In the year 2017, Medhi et al. had presented a quality evaluation framework that can be followed if web services are used for SaaS deployment [17]. In the year 2018, Gallardo et al. had proposed a SaaS adaptation framework that can be followed in industries. They had stated that the adaptation rate of SaaS can be raised by following well-established service policies. However, it lacks experimental results [18]. Most of the recent study was carried out on developing a strategy for evaluating service execution [19]. In the year 2018, Ochei et al. had presented a cross-case analysis of software architecture and process models that are primarily necessary for maintaining the trade-off associated with a business organization [20]. In the same year, Bora et al. had discussed the experimental arrangement and reliability assessment of multi service-oriented system for a higher load of usages [21]. The applicability of the method was validated through statistical analysis. In the year 2019, Kia et al. had proposed a novel hybrid methodology based on wireless network protocol [22]. They had stated that the performance of service execution can be enhanced through their proposed system. In the same year, Ahmed et al. had presented a novel implication methodology for comparing the SaaS performance through scalability measurement [23]. The applicability of the study was observed through a comparative model based on the experimental arrangement. Besker et al. (2019) had discussed a review on the importance of technical debt and time management policies for the effective delivery of the SaaS execution [24]. They had stated that the quality products of the system can be achieved through proper testing strategy. However, limited studies are observed that are carried out with experimental arrangements for SaaS execution in the multi-tenant and clustered web server. The novelty of the study can be observed in the way the SaaS is deployed with segregation of service layers and the evaluation of reliability while deploying it through load balancing clustered (LBC) and the nonLBC web server.

3. SYSTEM DESIGN AND METHODOLOGY

A novel system is designed and deployed by segregating the services of SaaS into the consumer layer, parent layer, service layer, and database layer. Each layer is responsible for doing a specific job. The consumer layer is the presentation layer for a graphical user interface (GUI). It contains design standards for the end-users. The parent layer works as a mediator between the consumer layer and the service layer. It is the broker for the service layer. It validates the communication received from multi-tenants. The service layer is the actual execution of the BL. The mathematical model that is required for functional operation is deployed here. The database layer is responsible for database transaction management. The database operation is handled by this layer. The four layers work together to satisfy the request received from multi-tenant machines. The architecture of the SaaS application is deployed in a server machine with clustered and non clustered configurations. The clustered web server has two working nodes, each containing the instances of the SaaS architecture. It also contains a load balancer that handles the request among working nodes. The load balancer periodically checks the existence of each

worker. If in any situation, a working node fails in processing instructions, the request of the end-user is redirected to other working nodes. Thus, the end-user will get a valid response from the SaaS application. The layers are kept independent of each other. As such, the coupling of modules is reduced. The consumer layer, parent layer and service layer is developed by using web service (WS) technology of JAX-WS with Java programming language. The database layer contains a database mapping of 15000 records of clinical health data. A mapping among disease and medicines is prepared for the system [25]. The multi-tenant of the SaaS is generated by using Mercury LoadRunner ersion 8.0 [26]. The tenants of the SaaS access the system simultaneously. Each execution set of tenants causes the invocation of HTTP request that in turn generates the Simple Object Access Protocol (SOAP) request for SaaS application. The testbed is configured by using the Mercury LoadRunner. It records the HTTP request made and failed during the execution of the SaaS application. The architecture of system design for SaaS application deployment is shown in Figure 1 below. The assessment framework for estimation of reliability of such deployment is developed and shown by using a flowchart in Figure 2.

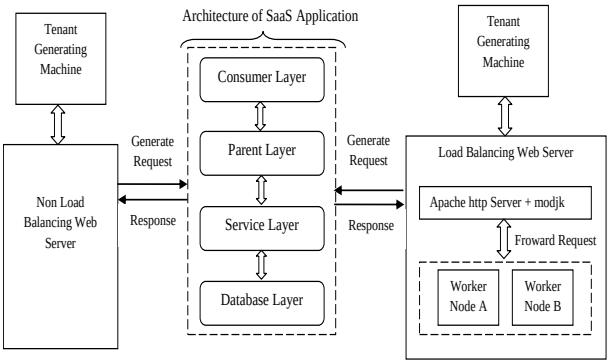

Figure 1. The architecture of system design for SaaS application deployment

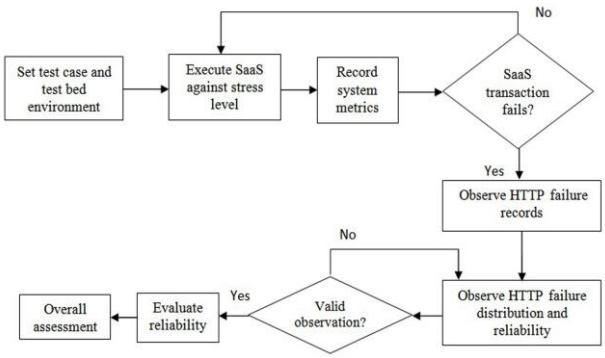

Figure 2. Flowchart for evaluating the reliability of SaaS using LBC and nonLBC web server

4. EVALUATION OF EXPERIMENTAL RESULTS

This section highlights the recorded system metrics by considering the HTTP request passed, failed and their failure rate against different stresses of usages while deploying SaaS by using clustered and non clustered web servers. The SaaS model is executed for the stress of usage for 50,100, 500, 800, 900, 1000, 1500, and 1800 system-generated virtual users (VU). A test case is developed that can retrieve data from the data store through a SaaS model. The system metrics such as total HTTP requests made along with the failed HTTP request out of the requested HTTP are recorded for the study. A user entry schedule of 1 user with a think time of 15sec is created by using a load testing tool Mercury Load Runner. The entire SaaS model is executed under this schedule. The schedule executes stress gradually over the SaaS model. Once all users enter the system, the test environment is set to run for 5 minutes for steady-state. Then all users exist from the test environment simultaneously. To observe how the SaaS behaves beyond its capacity, the system is executed up to 1800 user level. The recorded transaction responses are shown in Table 1. It is observed that with an increase in stress level, the HTTP request made is increasing gradually in both cases. However, the HTTP requests made by SaaS application by using LBC web server are observed to be comparatively higher than the LBC webserver. Since in LBC web server, the overhead of HTTP processing for specific BL is distributed among the clustered working nodes, for which the number of HTTP request that processed is sufficient for that stress level. In the case of SaaS in non LBC, it is observed that for the stress level of 900, a total of 48525 HTTP requests are generated and a total of 15251 HTTP requests are observed to be failed. The failure of the HTTP request is observed to be increasing gradually with SaaS in the non LBC web server.

TABLE 1. A comparative recorded metrics of SaaS using LBC and Non LBC web server (Total HTTP request: THR, Failed HTTP request: FHR, Failure Rate FHR/THR (%): FR)

Test Case	Stress Level	SaaS in Non LBC			SaaS in LBC		
		THR	FHR	FR	THR	FHR	FR
Data retrieval	50	366	0	0	284	0	0
	100	978	0	0	656	0	0
	500	14863	0	0	12598	0	0
	800	35924	0	0	19546	0	0
	900	48525	15251	0.31	47659	0	0
	1000	55082	17877	0.32	53275	0	0
	1500	121226	79650	0.65	62975	0	0
	1800	211675	170862	0.81	122616	42136	0.34

However, in the case of SaaS in the LBC web server, a total of 47659 HTTP requests are generated. In this case, no failure record is observed. For the stress level of 1800, the SaaS with the LBC web server has executed a total of 122616 HTTP requests. In this case, a total of 42136 HTTP requests are observed to be failed.

To observe the failure distribution of SaaS by using LBC and non LBC web server, a 30 repetitive execution of the test environment is performed. For the stress level of 900 users of SaaS in a non LBC web server, a data sample of 30 records is observed and evaluated. Similarly, the data sample is also prepared for the stress level of 1800 users of SaaS in the LBC web server. The histogram of the recorded data sample is prepared and shown in Figure 3 and Figure 4. Figure 3 depicts that for SaaS in a non LBC web server, the highest HTTP fault count lies with the ranges from 13171 to 17339. Similarly, Figure 4 depicts that for the LBC web server, the highest HTTP fault count lies with the ranges from 40529 to 43753. The histogram shows the existence of a single peak value in both cases.

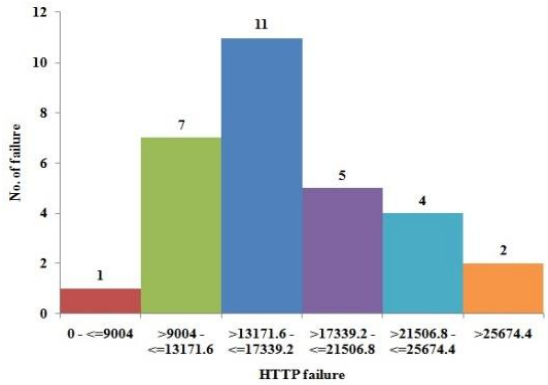

Figure 3. Histogram of fault count against 900 VU using SaaS in non LBC web server

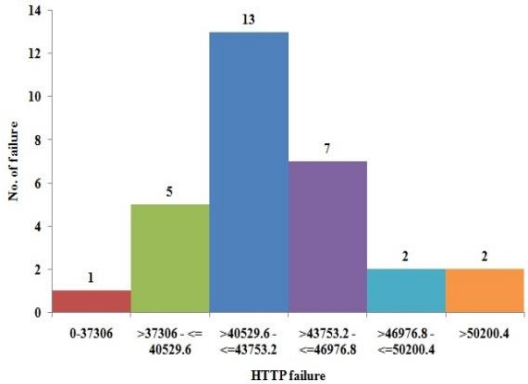

Figure 4. Histogram of fault count against 1800 VU using SaaS in LBC web server

As such, the normality of the failure record can be assumed. To better understand the distribution of recorded fault count, the normal probability plot (NPP) is drawn for both the cases.The NPP of data sample is shown in Figure 5 and Figure 6.

5. RELIABILITY OF SaaS

The reliability of SaaS is defined as the valid execution of the service during its execution period in a multi-tenant environment [27–29]. The reliability of SaaS by using LBC and nonLBC web servers is evaluated through the fault count model (FDM) [27]. From Table 1 it is observed that the SaaS execution in non LBC server is showing a valid response up to the execution of 800 tenants. However, the SaaS execution in the LBC server is showing valid response upto execution of 1500 tenants. As such, strong reliability can be assumed. However, the SaaS execution for 900 tenants by using non LBC server and 1800 tenants by using the LBC server, the reliability is estimated by using Equation (1).

$$R_{SaaS} = e^{-at} \quad (1)$$

Here, R_{SaaS} is the reliability of SaaS execution, ‘a’ is the rate of failure during the execution of SaaS in a multi-

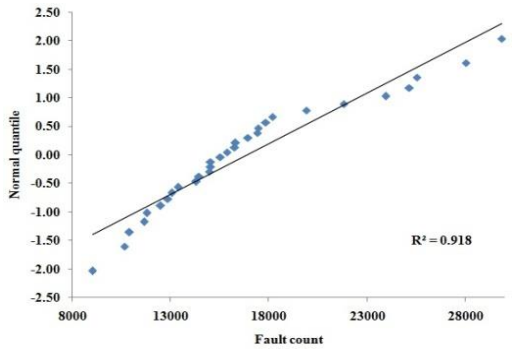

Figure 5. NPP of fault count against 900 VU using SaaS in non-LBC web server

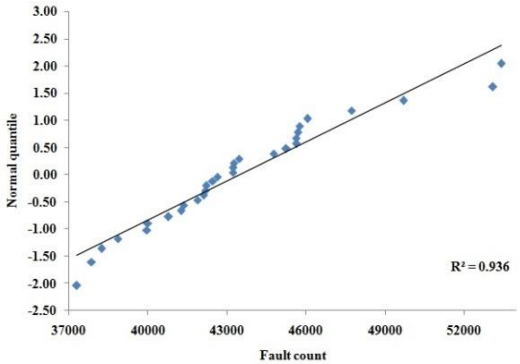

Figure 6. NPP of fault count against 1800 VU using SaaS in LBC web server

tenant environment and 't' is the observation time duration. In this study, the value of 't' is taken to be 1, as the observation of SaaS execution is made for one day. As such, for 900 tenants by using non LBC web server, the 'a' is recorded to be 0.31. For 1800 tenants by using LBC web server, the 'a' is recorded to be 0.34. Using Equation (1), R_{SaaS} for 900 tenants is evaluated to be 0.73 and R_{SaaS} for 1800 tenants is evaluated to be 0.71. As such, for SaaS by using non LBC web server, it will serve 73% of incoming request in a multi-tenant environment of 900 consumers. For SaaS by using the LBC web server, it will serve 71% of incoming requests in multi-tenant environment of 1800 consumers. In other cases, the SaaS application can generate an erroneous response to end-users.

6. OVERALL EVALUATION OF SaaS EXECUTION

The SaaS application is showing a valid response in a multi-tenant environment by using both clustered and non clustered webs server. The execution of SaaS in non LBC web server is observed to be stable up to 800 tenants of the service. However, in the case of SaaS with an LBC web server, it is stable upto 1500 tenants of service. Beyond that stress of usages level, the SaaS is generating invalid response against the incoming requests. For SaaS in non LBC web server, the failure records are observed at 900 tenants of the service. It had generated 48525 requests and out of them, 15251 requests were failed. However, in the case of SaaS with the LBC web server, it is recorded against 1800 tenants of the service. It had generated 122616 requests and out of them, 42136 requests were failed. The interpretation of histogram and NPP reveals the normality of the recorded fault count.The normal distribution of the SaaS application is observed in both cases. The strong reliability of SaaS can be achieved up to 800 and 1500 for non LBC and LBC web server, respectively. Beyond that, the reliability of SaaS degrades. For 900 tenants of the service with non LBC web server, the reliability of SaaS is estimated to be 0.73. For 1800 tenants of the service with an LBC web server, the reliability of SaaS is estimated to be 0.71. So, the moderate response of SaaS execution can be achieved in a multi-tenant environment. The overall evaluation and assessment of the SaaS execution in a multi-tenant environment predict that the SaaS with the LBC web server provides better reliability than SaaS without the LBC web server. The strong stability of service delivery can be achieved through SaaS with the LBC web server.

7. CONCLUSION

A novel methodology is proposed for estimating and evaluating the deployment of SaaS application by using non LBC and LBC web server. The applicability of SaaS execution is observed in both cases. The segregation of service layer for the deployment of SaaS application in the LBC web server imputes better reliability than its counterpart deployment by using a nonLBC web server. In a multi-tenant environment, the strong reliability for SaaS in the non LBC web server is recorded up to 800 simultaneous consumers. However, the strong reliability for SaaS in the non LBC web server is recorded up to 1500 simultaneous consumers. Both deployment techniques generate system failure after specific usages of tenants. However, the distribution of recorded failure is observed to be normal in both cases. Form the overall assessment it can be concluded that, with an increase in the stress level of usage, the rate of failure of SaaS with the LBC web server is comparatively lower than the SaaS without the LBC web server. A better reliable and stable information retrieval system can be achieved for a multi-tenant environment. However, in this work, the performance aspects of such deployment are not discussed. As part of our future work, we propose to study the quality of service for performance metrics of SaaS execution for a multi-tenant environment.

8. REFERENCES

1. Li, W., Zhang, Z., Wu, S. and Wu, Z., "An implementation of the SaaS level-3 maturity model for an educational credit bank information system", In 2010 International Conference on Service Sciences, IEEE, (2010), 283–287.
2. Jamshidi, R., "Bi-level Model for Reliability based Maintenance and Job Scheduling", ***International Journal of Engineering-Transactions C: Aspects***, Vol. 31, No. 3, (2017), 432–439.
3. Salajegheh, A. and Saberi, M., "Preventing Key Performance Indicators Violations Based on Proactive Runtime Adaptation in Service Oriented Environment", ***International Journal of Engineering - Transaction B: Applications***, Vol. 29, No. 11, (2016), 1539–1548.
4. Tsai, C.H., Ruan, Y., Sahu, S., Shaikh, A. and Shin, K. G., "Virtualization-based techniques for enabling multi-tenant management tools", In International Workshop on Distributed Systems: Operations and Management, Springer, (2007), 171–182.
5. El-Damcese, M., Abbas, F. and El-Ghamry, E., "Reliability Analysis of Three Elements in Series and Parallel Systems under Time-varying Fuzzy Failure Rate", ***International Journal of Engineering - Transaction C: Aspects***, Vol. 27, No. 4, (2014), 553–560.
6. Jacobs, D. and Aulbach, S., "Ruminations on multi-tenant databases", Datenbanksysteme in Business, Technologie und Web (BTW 2007), Germany, (2007), 1–5.
7. Nouri, S., "Automotive Vendor's Performance Evaluation and Improvement Plan Presentation by Using a Data Envelopment Analysis", ***International Journal of Engineering - Transaction B: Applications***, Vol. 31, No. 2, (2018), 374–381.
8. Chong, F. and Carraro, G., "Architecture strategies for catching the long tail", ***Microsoft Corporation***, Vol. 478, (2006), 9–10.
9. Saddik, A. E., "Performance measurements of web services-

based applications", ***IEEE Transactions on Instrumentation and Measurement***, Vol. 55, No. 5, (2006), 1599–1605.

10. Mietzner, R. and Leymann, F., "Generation of BPEL customization processes for SaaS applications from variability descriptors", In 2008 IEEE International Conference on Services Computing (Vol. 2), IEEE, (2008), 359–366.
11. Chen, X., A Service Quality Based Evaluation Model for SaaS Systems, Dissertation Thesis, University of Alberta, (2010).
12. Kang, S., Myung, J., Yeon, J., Ha, S.W., Cho, T., Chung, J.M. and Lee, S. G., "A general maturity model and reference architecture for SaaS service", In International Conference on Database Systems for Advanced Applications, Springer, (2010), 337–346.
13. Medhi, S. and Bezboruah, T., "Investigations on implementation of e-ATM Web Services based on NET technique", ***International Journal of Information Retrieval Research***, Vol. 4, No. 2, (2014), 41–56.
14. Candeia, D., Santos, R.A. and Lopes, R., "Business-driven long-term capacity planning for saas applications", ***IEEE Transactions on Cloud Computing***, Vol. 3, No. 3, (2015), 290–303.
15. Rahmani, M., Azadmanesh, A. and Siy, H., "Architectural reliability analysis of framework-intensive applications: A web service case study", ***Journal of Systems and Software***, Vol. 94, (2014), 186–201.
16. Sharif, S., Watson, P., Taheri, J., Nepal, S. and Zomaya, A. Y., "Privacy-aware scheduling SaaS in high performance computing environments", ***IEEE Transactions on Parallel and Distributed Systems***, Vol. 28, No. 4, (2016), 1176–1188.
17. Medhi, S., Bora, A. and Bezboruah, T., "Investigations on Evaluation of Some QoS Aspects of Service Oriented Computing System Based on Web Services", ***Sensors & Transducers***, Vol. 209, No. 2, (2017), 56–64.
18. Gallardo, G., Hernantes, J. and Serrano, N., "Designing SaaS for Enterprise Adoption Based on Task, Company, and Value-Chain Context", ***IEEE Internet Computing***, Vol. 22, No. 4, (2018), 37–45.
19. Mokhtari, H., Molla-Alizadeh, S. and Noroozi, A., "A reliability based modelling and optimization of an integrated production and preventive maintenance activities in flowshop scheduling problem", ***International Journal of Engineering - Transaction C: Aspects***, Vol. 28, No. 12, (2015), 1774–1781.
20. Ochei, L.C., Bass, J.M. and Petrovski, A., "Degrees of tenant isolation for cloud-hosted software services: a cross-case analysis", ***Journal of Cloud Computing***, Vol. 7, No. 1, (2018), 1–36.
21. Bora, A. and Bezboruah, T., "Some Aspects of Reliability Evaluation of Multi Service Multi-Functional SOAP Based Web Services", ***International Journal of Information Retrieval Research***, Vol. 8, No. 4, (2018), 24–38.
22. Kia, G. and Hassanzadeh, A., "HYREP: A Hybrid Low-Power Protocol for Wireless Sensor Networks", ***International Journal of Engineering - Transaction A: Basics***, Vol. 32, No. 4, (2019), 519–527.
23. Ahmad, A.A.S. and Andras, P., "Scalability analysis comparisons of cloud-based software services", ***Journal of Cloud Computing***, Vol. 8, No. 1, (2019), 1–17.
24. Besker, T., Martini, A. and Bosch, J., "Software Developer Productivity Loss Due to Technical Debt-A replication and extension study examining developers' development work", ***Journal of Systems and Software***, Vol. 156, (2019), 41–61.
25. Drug Index, Passi Publications, India, (2012), January-March.
26. Application-testing tool: Mercury LoadRunner 8.0, Available at: http://www.pcquest.com/pcquest/news/183659/application-testing-tool-mercury-loadrunner-80"
27. Medhi, S., Bora, A. and Bezboruah, T., "Investigations on some aspects of reliability of content based routing SOAP based windows communication foundation services", ***International Journal of Information Retrieval Research***, Vol. 7, No. 1, (2017), 17–31.
28. Wang, L., Bai, X., Zhou, L. and Chen, Y., "A hierarchical reliability model of service-based software system", In 2009 33rd Annual IEEE International Computer Software and Applications Conference (Vol. 1), IEEE, (2009), 199–208.
29. Shooman, M.L., Reliability of computer systems and networks: fault tolerance, analysis, and design, John Wiley & Sons, (2003).

Investigation on Reliability Estimation of Loosely Coupled Software as a Service Execution Using Clustered and Non-Clustered Web Server

A. Bora, T. Bezboruah

Department of Electronics & Communication Technology, Gauhati University, Guwahati, India

P A P E R I N F O

Paper history:
Received 20 May 2019
Received in revised form 01 October 2019
Accepted 08 November 2019

Keywords:
Clustered Web Server
Software as a Service
Multi Tenant Environment
Reliability

چکیده

ارزیابی قابلیت اطمینان نرم‌افزارها کاملاً همراه به عنوان یک سرویس از طریق پارادایم یک سرور وب مبتنی بر خوشه‌ای و غیر خوشه‌ای، یک ویژگی مهم برای ارائه و اجرای سرویس محسوب می‌شود. ما یک روش جدید برای اندازه‌گیری قابلیت اطمینان نرم‌افزار به عنوان اجرای سرویس از طریق تست بار پیشنهاد نمودیم. تعداد گسل‌های مدل در برابر استرس کاربران مستقر شده است. یک برنامه نمونه اولیه با استفاده از سرویس وب ساده مبتنی بر پروتکل دسترسی به شی ساده ساخته شده است و مطالعه در آنجا انجام می‌شود. مجموعه آزمایشی، معماری، نتایج آزمایش بار و به دنبال آن یک مطالعه مقایسه ای در این کار بحث شده است. قابلیت اطمینان سرویس با استفاده از سرور وب خوشه‌ای و غیر گروه‌بندی شده پس از حد مشخصی از نقطه اجرای سطح استرس کاهش یافته مشاهده گردید. بعلاوه قابلیت اطمینان سرویس با استفاده از یک وب سرور مبتنی بر خوشه بهتر از سرویس‌های دارای یک سرور وب غیر خوشه‌ای است ارزیابی تطبیقی پیش‌بینی می گردد. با افزایش سطح استرس استفاده در یک محیط چند مستأجر، سرویس با وب سرور خوشه‌ای قابلیت اطمینان بهتری نسبت به این سرویس بدون یک سرور وب خوشه‌دار فراهم می‌کند. وقوع عدم موفقیت پروتکل انتقال پروتکل Hyper Text در سرویس با یک وب سرور خوشه‌ای نسبتاً کمتر از سرویس همتای خود بدون داشتن سرور وب خوشه‌ای است. این مطالعه به شناسایی کاربردی بودن روش کمک می‌کند و اثربخشی چنین استقرار را نشان می‌دهد.

***doi**: 10.5829/ije.2020.33.01a.09*