Asian Journal of Information Technology 15 (13): 2228-2237, 2016
ISSN: 1682-3915

Creation of Software Testing Environment in Cloud Platform

J. Frank Vijay and B. Hariharan
Department of Information Technology, KCG College of Technology, Chennai, India

Abstract: Cloud computing is a novel computing standard that provides major support for the software testing and development. In this study, an efficient software testing framework with weight-based prioritization technique is proposed for performing software testing in the distributed cloud environment. At first, thetest case dataset is initialized then the frequent test case in the dataset is estimated. The weight of the similar test cases are determined using the cosine similarity. With the estimated weight values, the weight based prioritization technique is applied for prioritizing the test cases. After prioritization, the K-Medoid clustering algorithm is deployed for clustering the similar test cases. To provide security for the test cases, an Attribute key based Elliptic Curve Cryptography (ECC) algorithm is used for encryption and decryption. Further, a cache memory is exploited for optimizing the memory consumption of the test case execution. The proposed framework is deployed in cloudsim and the experimental results prove that the proposed framework provides optimal results than the existing random, prioritized techniques, K-mean algorithm, Hierarchical algorithm, Test Case Prioritization (TCP) technique, Adaptive Random Test case prioritization (ART), Local Beam Search (LBM) and greedy approaches.

Key words: K-medoid, attribute-based Elliptic Curve Cryptography (ECC), cosine similarity, weight-based prioritization technique, Test Case Prioritization (TCP) technique, Adaptive Random Test case prioritization (ART), Local Beam Search (LBM)

INTRODUCTION

One of the important phases of the software development life cycle is software testing. Testing a software demands expensive dedicated infrastructures and resources for analyzing the following (Narula and Sharma, 2014):

- Application performance
- Reliability
- Security
- Functionality
- Speed

As the business requirement increases, the organization find difficult to maintain an in-house testing facilities that imitate the real-time environments. To address this issue one of the popular distributed computing approach named cloud computing is used. The exploitation of cloud computing environment for the software testing and maintenance has the advantages such as enhanced security, minimal cost requirement for the testing process, improved testing efficiency and realistic performance testing (Jagadeesh, 2012).

Motivation: Distributed cloud is a cloud computing technology that interconnects the nodes located in various geographic locations. The existing systems exploit the clustering algorithms such as agglomerative clustering and K-means algorithm for providing an efficient software testing. But in agglomerative clustering the distance between two clusters are always greater. Further, it is sensitive to noise, outliers, breaks the large clusters and finds difficult to handle the variable sized clusters. As the K-means clustering algorithm has fixed number clusters, it is difficult to predict the K-value. Hence, to address the issues in the existing clustering algorithms, a cosine similarity based K-Medoid algorithm is proposed. The K-Medoid algorithm constructs a similaritymatrix for clustering the test cases with similar values. As the test case clusters are similar, accurate results are achieved on executing the test case.

To protect the data from cloud environment attacks, the cryptographic algorithms such as AES and DES are commonly used. But, the level of security provided by these algorithms are not satisfactory. Hence, to enhance the security of the data, an efficient attribute key based ECC algorithm is proposed for the encryption process. When compared to the traditional cryptographic algorithms, the proposed algorithm increases the level of security using key pair based cryptography calculation. Further, the use of public and private key for the encryption process increases the security than AES and DES algorithms. The generated keys are maintained in the

Corresponding Author: J. Frank Vijay, Department of Information Technology, KCG College of Technology, Chennai, India

hash table for faster retrieval. Both the encryption and decryption processes exploit the same key for encrypting and decrypting the clustered test cases. The execution of the test cases exploit a cache memory. Failure in the test case execution removes the test case from the cache memory for optimizing the memory space. The results from each node in the distributed cloud environment are merged for providing the final test report.

Objectives: The key objectives of the proposed software testing framework are listed below:

- To deploy weight-based prioritization technique for prioritizing the test cases
- To implement cosine similarity based K-Medoid clustering algorithm for clustering the test cases after prioritization
- To enhance the security of test case execution in cloud environment using attribute key based ECC algorithm
- To optimize the memory consumption of each test case execution using cache memory

Literature review: This study provides an illustration of the existing test case prioritization techniques, clustering techniques data security techniques and security mechanisms used for the cloud computing environment.

Test case prioritization techniques: Parthiban *et al.* (2014) suggested a dependency estimation based test case prioritization technique for estimating the dependencies between the tests. The information from the previous iteration was not demanded for estimating the priorities. Further, the fine-grained test suites were not maintained. The functional dependency, precision, recall and F-measure were estimated using theranking algorithm. The functional estimation was based on the number of accurate functions in the application. When compared to the random and untreated testing suites, the proposed technique provided optimal error detection. Miller (2013) proposed a family of test case prioritization techniquesfor prioritizing the test suite. The dependencies in the test ordering were preserved. The evaluation results proved that the proposed technique automatically extracted the dependency structures from the test suites. The disadvantage of the proposed technique was the lack of assumption of the dependency relationship between the test cases (Nivethitha and Sriram, 2013) suggested a Dependency Structure Prioritization (DSP) for prioritizing the jobs in the cloud environment. By reducing the number of job migration and missed deadline jobs the resource scheduling was improved. Srikanth *et al.* (2016) suggested an efficient prioritization scheme for addressing the schedule and budget constraints at the testing phase. The suggested scheme exploited the risk information of the system and developed a two-requirement based TCP approach. (Nguyen *et al.*, 2011) suggested a prioritization approach for validating the test cases using the information retrieval. Based on the changes in the service, the test cases were prioritized. Roongruangsuwan and Daengdej (2010) surveyed multiple test case prioritization techniques. Further, the issues such as ignoring practical weight factors, inefficient test case prioritization method, ignoring the test case size were analyzed. From the results obtained from the analysis, the ability of the weight was improved and the rank test cases was enhanced using the practical factors. Hettiarachchi *et al.* (2016) suggested a fuzzy expert system for providing an efficient test case prioritization. The suggested system exploited the requirement modification status, security, complexity and size of the software requirements as risk indicators. Further, a fuzzy expert system was proposed for estimating the risk requirements. The advantages of the proposed system were early detection of faults andflexible application of the systematic approach to the industrial applications. Solanki *et al.* (2015) proposed a modified Ant Colony Optimization (m-ACO) technique for prioritizing the test cases. The performance of the suggested m-ACO was validated with the average percentage of the faults detected. Anitha and Srinath (2014) surveyed the various prioritization, clustering, load balancing and security techniques for the cloud environment. From the survey results it was found that the Dependency Structure Prioritization (DSP) techniquewas optimal for prioritizing the test cases. The DSP technique reduced the time consumption and also maximized the fault rate prediction. The clustering of the test cases using agglomerative clustering was found to be more flexible in the level of granularity. The exploitation of the resources were optimal for hierarchical load balancing algorithm. The enhancement of the cloud security using Diffie-Hellman algorithm was found to be more efficient that the other existing algorithm.

Clustering techniques: Srivastava *et al.* (2013) proposed a hierarchical agglomerative clustering algorithm for maximizing the execution efficiency of the tasks in the distributed cloud computing environment. Experimental results proved that the proposed clustering algorithm decreased the execution time, increased the efficiency and parallelism. Kaur and Kaur (2013) proposed a query redirection method for enhancing the performance and accuracy of the K-means clustering algorithm. By

exploiting the validationmeasures such as entropy, time, f-measure and coefficient of variance, the performance of the k-means clustering algorithm and hierarchical clustering algorithm were analyzed. The analysis results proved that the K-means algorithm produced less execution time and optimal performance than the hierarchical algorithm. The disadvantage of the suggested analysis was the lack of consideration of normalized and zun-normalized data. Cui *et al.* (2014) suggested a K-means clustering algorithm and a processing model for preventing the iteration dependence. Experimental analysis proved that the proposed model enhanced the efficiency, robustness, scalability and performance. Celebi *et al.* (2013) analyzed the various clustering algorithms along with their computational efficiency. Multiple performance criteria was used for comparing the linear time complexity on a large and diverse collection of the datasets. By performing the non-parametric statistical tests, the experimental results using non-parametric statistical tests were analyzed.

Data security techniques: Ora and Pal (2015) suggested a combination of Rivest-Shamir-Adleman partial homomorphic and MD5 hashing algorithm for maintaining the data security and data integrity in cloud environment. Before uploading the data into the cloud server, it was encrypted using RSA partial. After uploading the data to the cloud server its hash value was estimated using MD5 hashing scheme. Li *et al.* (2013) proposed an Attribute-Based Encryption (ABE) technique for encrypting the Personal Health Record (PHR) file. Based on the multiple data owner scenario, the users of the PHR file were divided into multiplesecurity domains. The exploitation of the multi-authority ABE enhanced the privacy degree of the patients. Further, the scalability, security and efficiency of the proposed ABE technique were increased. Sood (2012) proposed a framework that includes various techniques and specialized procedures for protecting the data from the owner to the cloud and then to the user. The data was classified using three parameters such as Confidentiality (C), Availability (A) and Integrity (I). The classified data was encrypted using Secure Socket Layer (SSL) 128 bit encryption. The number of bits was then increased to 256 bit. The integrity of the data was validated using Message Authentication Code (MAC). Experimental analysis proved that the proposed framework achieved optimal reliability, availability and integrity.

Security mechanisms: Rewagad and Pawar (2013) suggested a hybrid of digital signature and Diffie Hellman exchange blended with AES algorithm for enhancing the confidentiality of the data stored in cloud. As the proposed algorithm was composed of three-way mechanism, the hackers were unable to crack the system security. Malik and Kumar (2015) suggested a data protection model for securing the data from the cloud environment attackers. The proposed model encrypted the data using AES algorithm and authenticated the data using Diffie Hellman algorithm. The suggested model prevented the key computation overhead and management overhead. Kumar *et al.* (2012) suggested an ECC algorithm for protecting the data files in the cloud. The suggested algorithm permitted only the group members to access the data stored over the shared data section. Rewagad and Pawar (2013) proposed a hybrid of digital signature and Diffie Hellman key exchange algorithms for enhancing the confidentiality of the data stored in cloud. The proposed algorithm prevented the hackers from cracking the security and integrity of the system and thereby it protected the data stored in the cloud. As the proposed model integrated the AES and 3DES, the algebraic attack on the hybrid model was minimized. Dubey *et al.* (2012) suggested aRSA and MD5 based algorithm for providing cloud-user security. After the cloud user uploads the data onto the cloud environment, the data was encrypted using RSA algorithm. The decryption of the uploaded data was performed using the private key of the cloud admin. The data was updated using a secure key with message digest tag.

From the analysis of the various test case prioritization techniques, clustering techniques, data security techniques and security mechanisms it is found that the existing techniques consume more execution time for the prioritization process and consumes more computation time for the encryption and decryption processes. Further, the existing techniques do not consider the memor optimization. Hence, to overcome all these limitations, a secure software testing using weight-based prioritization technique is proposed.

MATERIALS AND METHODS

Software testing framework with weighted prioritization technique: This section provides a detailed explanation regarding the proposed software testing framework with weighted prioritization technique. The overall flow of the proposed software testing framework is shown in Fig. 1.

Generation of test case: Initially, a distributed cloud environment is created with 'N' number of nodes, then a test case dataset 'D' is initialized. Each test case 'T' in the dataset is allocated a test case ID. The steps involved in the test case generation are illustrated:

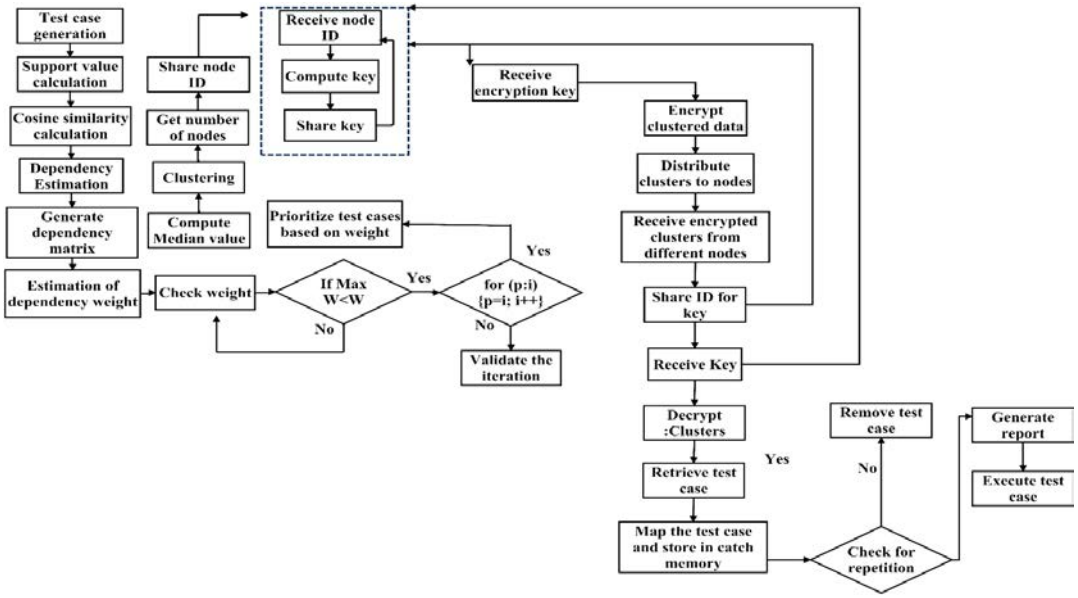

Fig. 1: Overall flow of the proposed software testing framework with weighted prioritization technique

Algorithm A; Test case initialization input: test report (D):

Output:Test case specification (T)
Step 1: D = Test report dataset
Step 2: TID = Test case ID
Step 3: T_i = Test case specification for the i^{th} test case i = 0, 1, ... n

Calculation of support value and cosine similarity: After the initialization of the test cases, the support value for the test cases are calculated by finding the frequent test case in the dataset. To determine the weight of the similar test cases, the cosine similarity is proposed. The following are the steps involved in the cosine similarity calculation.

Algorithm B; Cosine similarity calculation:

Input: Test case specifications (T) in the dataset 'D'
Output:Support value for each transaction
For each Test case $T_i \in D$
For each Test case $T_j \in D$

$$\cos\theta = \frac{Sup(T_i) * Sup(T_j)}{|Sup(T_i)| * Sup(T_j)} \quad (1)$$

End for
End for
//weight calculation
//Sort test case based on weight value
Sort ($T_i \in D$)
For each Test case $T_i \in D$
Weight (T_i) = Find mean (T_i, D)
End for
//Prioritize the Test case
Apply priority based on Weight value

With the estimated cosine similarity values, an adjacency matrix is constructed using (Eq. 1) and the mean of all the similarity values is used for calculating the weight of each test case. Based on the calculated weight, the weight-based prioritization technique is applied. The suggested technique prioritize the test cases based on the weight value. The advantage of the proposed prioritization technique is increased fault tolerance rate than the traditional prioritization techniques.

Clustering using K-medoid algorithm: After the prioritization process the number of nodes in the distributed environment is computed as 'N', then the computed number is initialized as the number of clusters as follows:

$$\text{Cluster } k = N$$

Based on the number of nodes, the test case from the matrix is clustered. With respect to the number of nodes, the test case prioritization size is estimated as follows:

$$P = size(D)/k \quad (2)$$

Where:
D = Denotes the dataset
k = Represents the number of clusters

For each cluster O_1, ..., O_k and for each test case $T_i \in P$, the clustering is performed using K-Medoid algorithm. The K-Medoid algorithm exploits the most central object named Medoid of eachcluster. The main aim of the K-Medoid algorithm is to estimate the clusters O_1, O_2, ..., O_k that minimizes the following target function,

$$\sum_{a=1}^{k} \sum b \in L_i d(b, j_b) \quad (3)$$

Where:

k = Denotes the number of clusters
L_i = Denotes the cluster
$d(b, j_b)$ = Represents the target function

Algorithm C; K-Medoid algorithm:

Input: Number of clusters (k), Dataset D with m objects
Output: Set of k clusters that reduces the dissimilarities of all the objects
Step 1:k number of objects such as O_1, .., O_k are chosen as the initial Medoids.
Step 2: Compute the value of the target function using (Eq. 3)
Step 3: For all (O_i, f_3) pairs the target function is improved by moving the f_3 non-medoid point to a new medoid point and moving the medoid point O_i to a new non-medoid point.

The proposed algorithm is repeated till k number of clusters areformed. The following advantages are the reasons for choosing the K-Medoid clustering algorithm.

Advantages of K-Medoid algorithm:

- Minimal execution time for clustering process
- Robust
- Easy to implement
- Provides clustering result in minimal number of steps
- Prevents the outliers

After clustering the similar test cases, the created clusters are distributed among various nodes of the distributed cloud environment. To facilitate a secure cluster distribution, the clusters are forwarded in an encrypted format.

Encryption and decryption using ECC: To protect the test cases from the security threats of the cloud, an Attribute key based Elliptic Curve Cryptography algorithm is proposed. Initially, the attribute key is generated as follows:

For each node N_i... N_n

$$AK(N_i) \leftarrow NodeId(N_i) \quad (4)$$

End For

The suggested attribute based ECC algorithm assigns the node ID of a nodeas the attribute key of the same node. After the generation of the attribute key it is stored in the Hash table with the corresponding node information. After the generation of the attribute key, the ECC algorithm is used for encrypting and decrypting the clusters. The steps involved in the ECC based cluster encryption is illustrated as follows.

Algorithm D: Encryption using ECC algorithm:

Step 1: Initialize the private and public key using Eq. 5 and 6:
For each cluster C_i ... C_n

$$\text{Private Key} = AK(N_i) \quad (5)$$

$$\text{Public Key} = d^*\text{Private Key} \quad (6)$$

Where:
d = A random value
Step 2 = Initialize two cipher text as C1 and C2
Step 3 = Calculate C1 using Eq. 7

$$C_1 = k^*P \quad (7)$$

Where:
k = Denotes a random number
p = Represents the number of partitions

Step 4: Compute C2 using Eq. 8

$$C_2 = M+^*Q \quad (8)$$

Where:
M = Denotes the message
k = Represents the random number
Q = Denotes the public key

Step 5: After encrypting the clustered data, the encrypted cluster is mapped to the nodes as follows:
For each encrypted cluster E (C_i)...E (C_n)
For each node N_i ... N_n:

$$N_i \leftarrow E(C_i) \quad (9)$$

End for
End for

The proposed attribute based ECC algorithm creates the private and public key using two cipher text such as C1 and C2. The message that needs to be encrypted is denoted as 'M'. Both the text C1 and C2 are campaigned and assigned for each node created on the distributed environment. The generated keys are maintained in the hash table for faster retrieval. To decrypt the test cases the keys that are used for the encryption process are used. The steps that are involved in the decryption process is illustrated below:

Algorithm E; Decryption using ECC algorithm:

For each encrypted cluster E (C_i) ... E (C_n)
Get AK from the distributed hash table
D*(C_i)<-Decrypt (E (C_i))
Private Key = ABK (N_i)

$$\text{Public key->}d^*\text{Private Key} \quad (10)$$

$$M = C2\text{-}d^*C1 \quad (11)$$

End for

At first, the attribute key for each encrypted cluster is retrieved from the distributed hash table then the encrypted cluster is decrypted and assigned toD (C_i). The attribute key for each node is assigned as the private key and the public key is estimated based on Eq. 10. After the estimation of the private and the public key, the message is decrypted using (Eq. 11).

Memory optimization: The test cases are executed using the cache memory. If any test case gets failed on their execution it is removed from the cache memory, thus optimizing the memory space. The overall steps involved in the memory optimization process is illustrated below.

Algorithm F; Steps involved in the memory optimization:

Step 1: Initialization of the cache
    For each Cluster C_i .... C_n
    Cache <- Execute (C_i)
    End for
Step 2: Removal of failed test cases
    Cache. remove (failed test case)
Return cache

RESULTS AND DISCUSION

Performance analysis: This section illustrates the performance results of the proposed software testing framework with weighted prioritization technique. The following are the metrics used for comparing the performance of the proposed method:

- Percentage of defect detected
- Percentage evaluation of test case
- Entropy
- Execution time
- Test case execution time
- Prioritization time

Table 1: Comparison of percentage evaluation of test case

Metrics	Random (%)	Prioritized (%)	Proposed support based weight prioritization (%)
Data severity	51	72	88
Test case size	75	80	91
Total prioritization time	50	30	10

Percentage of defect detected: The number of defects estimated during each test case executionis denoted as percentage of defect detection. By detecting the defects, the repeated test cases are eliminated. The percentage of defect detected with respect to the percentage of the test cases execution for the existing random and prioritized technique (Muthusamy, 2013) and the proposed support weight based prioritization is depicted in Fig. 2. The comparison results show that the proposed supportweight based prioritization increases the percentage of the test cases execution than the existing techniques.

Percentage evaluation of test case: To evaluate the test cases the metrics such as data severity, test case size and total prioritization time are used. The performance of the proposed support weight based prioritization technique for these metrics are compared with the existing random and prioritized (Muthusamy, 2013) techniques. The comparison results are depicted in Table 1. From Table 1 it is clear that the proposed support weight based prioritization technique provides optimal performance than the existing techniques.

Entropy: Entropy measures the purity of the clusters with respect to the class labels. The lower the value of the entropy, the higher will be the quality of the clustering. The comparison of entropy for the existing K-means, Hierarchical clustering algorithms (Kaur and Kaur, 2013) and the proposed weight based K-Medoid algorithm is

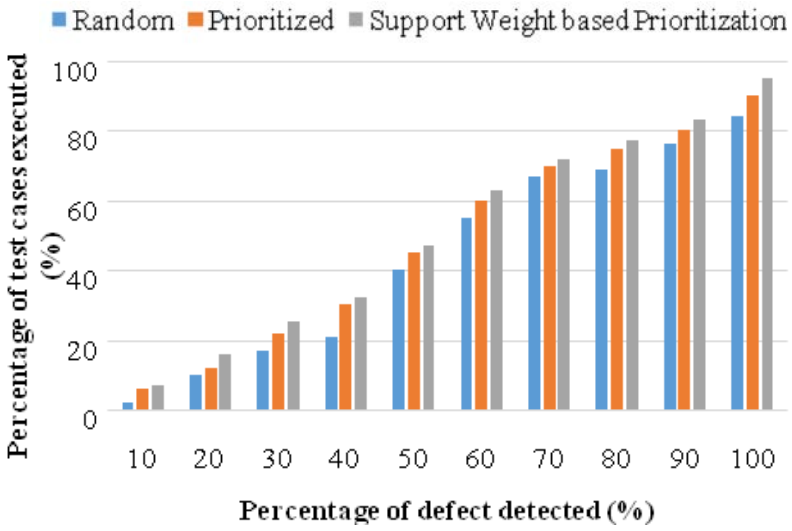

Fig. 2: Comparison of percentage of defect detected for the existing and the proposed methods

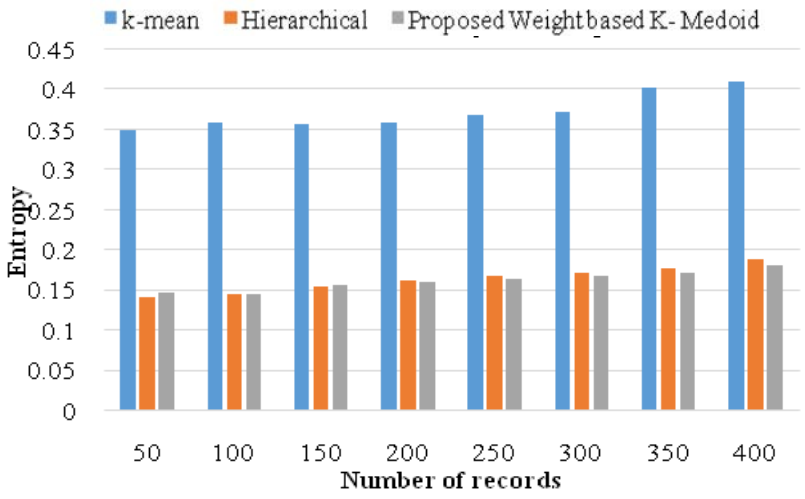

Fig. 3: Comparison of entropy for the existing and the proposed methods

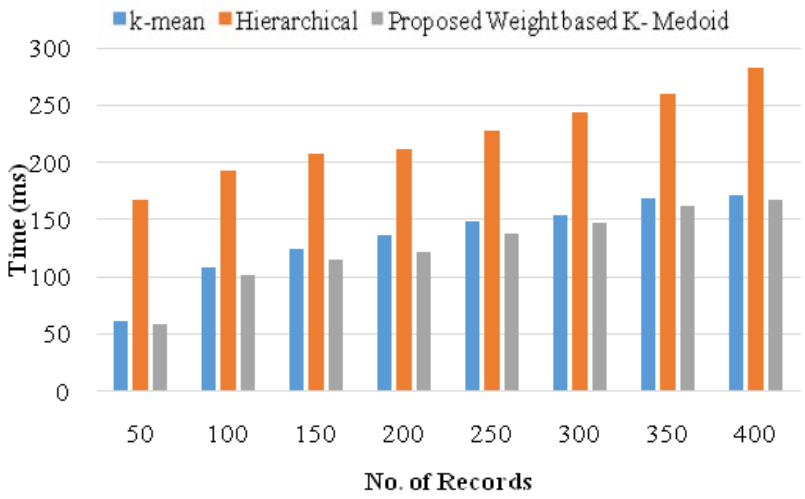

Fig. 4: Comparison of execution time for the existing and the proposed methods

depicted in Fig. 3. The analysis results show that the proposed algorithm provides an optimal entropy value than the existing algorithms.

Execution time: The amount of time consumed for the execution of different number of records is depicted in Fig. 4. The performance of the proposed weight based K-Medoid algorithm is validated with the existing algorithms such as K-Mean algorithm, hierarchical clustering algorithm (Kaur and Kaur, 2013). The comparison results show that the proposed clustering algorithm provides minimal execution time than the existing K-means and hierarchical clusteringalgorithm.

Test case execution time: The amount of time consumed for the execution of each test case is shown in Fig. 5. To validate the performance of the proposed support weight based prioritization, it is compared with the existing Test Case Prioritization (TCP) technique (Elbaum *et al*., 2002). The validation results prove that the proposed support weight based prioritization technique provides minimal execution time for all the test cases.

Prioritization time: The prioritization time is defined as the amount of time consumed for the prioritization process.To validate the performance of the proposed support weight based prioritization, it is compared with the existing techniques such as Adaptive Random Test case prioritization (ART), Local Beam Search (LBM) and greedy approaches (Jiang and Chan, 2015). The comparison results shown in Fig. 6 shows that the proposed support weight based prioritization method provides minimal prioritization time than the existing ART, LBM and greedy approaches.

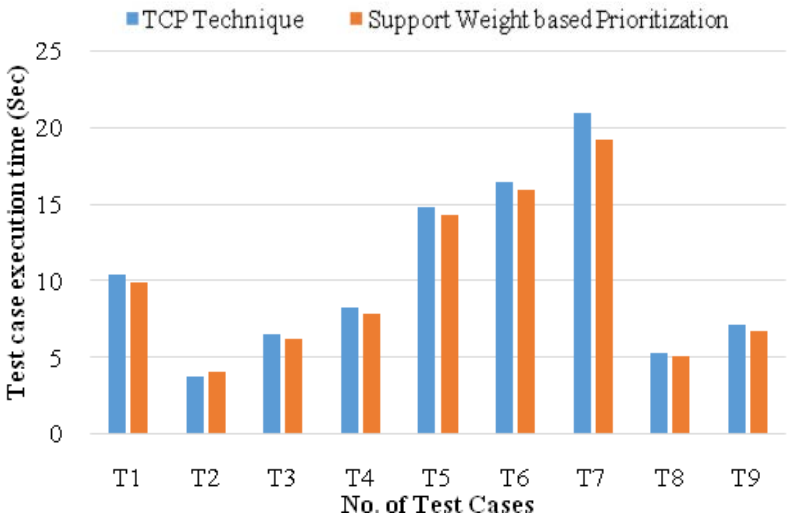

Fig. 5: Comparison of test case execution time for the existing TCP technique and support weight based prioritization

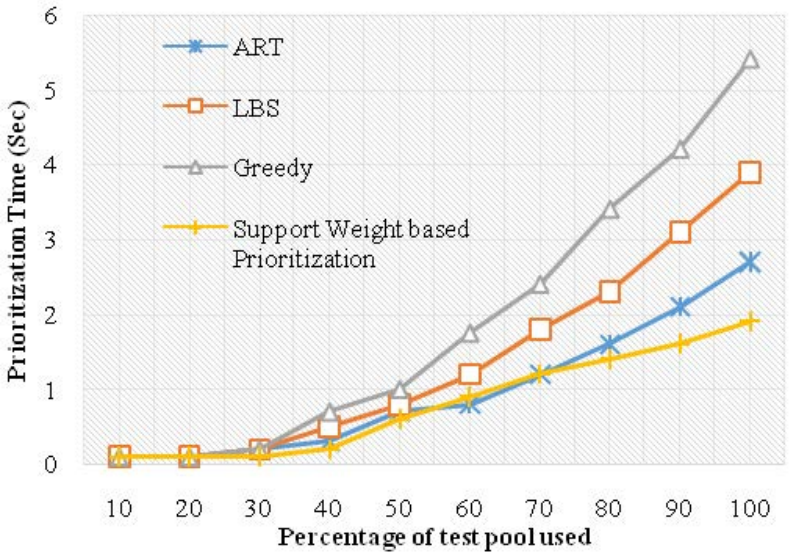

Fig. 6: Comparison of prioritization time for the existing and the proposed methods

CONCLUSION

Software testing is the process of validating the performance of the software package functions according to the user expectations. As cloud computing environment provides the services to the customers in a flexible manner it is preferred for the software testing process. In this study, an efficient and secure weight-based prioritization technique is proposed for performing the software testing in distributed cloud environment. Initially, a distributed cloud environment is created with 'N' number of nodes then the test case dataset is initialized. The suggested technique exploits the cosine similarity for estimating the weights of the similar test cases. Based on the weight values, the prioritization technique is applied for prioritizing the test cases. After the prioritization process, the K-Medoid algorithm is proposed for clustering the similar test cases.As the test cases are executed in the open environment of cloud, it is prone to various security threats. Hence, to enhance the security of the test cases an attribute based ECC algorithm is proposed for the encryption process. To provide faster retrieval of keys, the keys are stored and retrieved from the hash table. By exploiting the cache memory the memory consumption for the testing process is reduced. Finally, the results from all the nodes are

merged and the test report is produced as output. To validate the performance of the proposed secure weight-based prioritization technique it is compared with the existing random and prioritized technique for the metrics such as percentage of defect detected, percentage of evaluation of test case. The entropy and execution time for the proposed technique are compared with the existing K-means, hierarchical algorithms. The test case execution time of the proposed method is validated with the existing TCP technique. Further, the prioritization technique of the proposed technique is compared with the existing ART, LBS and greedy techniques. The comparison results show that the proposed method provides optimal performance than the existing methods.

As a future enhancement the security of the test cases can be enhanced. Further, the key size required for the computation can be made smaller and the efficiency of the prioritization process can be increased.

REFERENCES

Anitha, D. and M.V. Srinath, 2014. A review on software testing framework in cloud computing. Int. J. Comput. Sci. Inf. Technol., 5: 7553-7562.

Celebi, M.E., H.A. Kingravi and P.A. Vela, 2013. A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Syst. Appl., 40: 200-210.

Cui, X., P. Zhu, X. Yang, K. Li and C. Ji, 2014. Optimized big data k-means clustering using MapReduce. J. Supercomputing, 70: 1249-1259.

Dubey, A.K., A.K. Dubey, M. Namdev and S.S. Shrivastava, 2012. Cloud-user security based on RSA and MD5 algorithm for resource attestation and sharing in java environment. Proceedings of the 2012 CSI Sixth International Conference on Software Engineering (CONSEG), September 5-7, 2012, IEEE, Indore, India, ISBN: 978-1-4673-2174-7, pp: 1-8.

Elbaum, S., A. Malishevsky and G. Rothermel, 2002. Test case prioritization: A family of empirical studies. IEEE Trans. Software Eng., 28: 159-182.

Hettiarachchi, C., H. Do and B. Choi, 2016. Risk-based test case prioritization using a fuzzy expert system. Inf. Software Technol., 69: 1-15.

Jagadeesh, S.A., 2012. Cloud based testing: Need of testing in cloud infrastructures and cloud platforms. Int. J. Comput. Sci. Inf. Technol. Secur., 2: 398-401.

Jiang, B. and W.K. Chan, 2015. Input-based adaptive randomized test case prioritization: A local beam search approach. J. Syst. Software, 105: 91-106.

Kaur, M. and U. Kaur, 2013. Comparison between K-mean and hierarchical algorithm using query redirection. Int. J. Adv. Res. Comput. Sci. Software Eng., 3: 1454-1459.

Kumar, A., B.G. Lee, H. Lee and A. Kumari, 2012. Secure storage and access of data in cloud computing. Proceedings of the 2012 International Conference on ICT Convergence (ICTC), October 15-17, 2012, IEEE, Jeju City, South Korea, ISBN: 978-1-4673-4829-4, pp: 336-339.

Li, M., S. Yu, Y. Zheng, K. Ren and W. Lou, 2013. Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. IEEE Trans. Parallel Distrib. Syst., 24: 131-143.

Malik, R. and P. Kumar, 2015. Cloud computing security improvement using Diffie Hellman and AES. Int. J. Comput. Appl., 118: 25-28.

Miller, T., 2013. Using dependency structures for prioritization of functional test suites. IEEE. Trans. Software Eng., 39: 258-275.

Muthusamy, S.K.T., 2013. A test case prioritization method with weight factors in regression testing based on measurement metrics. Int. J. Adv. Res. Comput. Sci. Software Eng., 3: 390-396.

Narula, T. and G. Sharma, 2014. Framework for analyzing and testing cloud based applications. Int. J. Adv. Res. Comput. Sci. Software Eng., 4: 592-596.

Nguyen, C.D., A. Marchetto and P. Tonella, 2011. Test case prioritization for audit testing of evolving web services using information retrieval techniques. Proceedings of the 2011 IEEE International Conference on Web Services (ICWS), July 4-9, 2011, IEEE, Washington, DC., USA., ISBN: 978-1-4577-0842-8, pp: 636-643.

Nivethitha, S. and V.S. Sriram, 2013. Consolidating batch and transactional workloads using dependency structure prioritization. Int. J. Eng. Technol., 5: 128-1334.

Ora, P. and P.R. Pal, 2015. Data security and integrity in cloud computing based on RSA partial homomorphic and MD5 cryptography. Proceedings of the 2015 International Conference on Computer, Communication and Control (IC4), September 10-12, 2015, IEEE, Indore, India, ISBN: 978-1-4799-8163-2, pp: 1-6.

Parthiban, D.T., M.R. Kamalraj and D.S. Karthik, 2014. Establishing a test case prioritization technique using dependency estimation of functional requirement. Int. J. Inn. Res. Sci. Eng. Technol., 3: 1526-1529.

Rewagad, P. and Y. Pawar, 2013. Use of digital signature with Diffie Hellman key exchange and AES encryption algorithm to enhance data security in cloud computing. Proceedings of the 2013 International Conference on Communication Systems and Network Technologies (CSNT), April 6-8, 2013, IEEE, Gwalior, India, ISBN: 978-1-4673-5603-9, pp: 437-439.

Roongruangsuwan, S. and J. Daengdej, 2010. A test case prioritization method with practical weight factors. J. Software Eng., 4: 193-214.

Solanki, K., Y. Singh and S. Dalal, 2015. Test case prioritization: An approach based on modified ant colony optimization (m-ACO). Proceedings of the 2015 International Conference on Computer, Communication and Control (IC4), September 10-12, 2015, IEEE, Indore, India, ISBN: 978-1-4799-8163-2, pp: 1-6.

Sood, S.K., 2012. A combined approach to ensure data security in cloud computing. J. Network Comput. Appl., 35: 1831-1838.

Srikanth, H., C. Hettiarachchi and H. Do, 2016. Requirements based test prioritization using risk factors: An industrial study. Inf. Software Technol., 69: 71-83.

Srivastava, K., R. Shah, D. Valia and H. Swaminarayan, 2013. Data mining using hierarchical agglomerative clustering algorithm in distributed cloud computing environment. Int. J. Comput. Theory Eng., 5: 520-522.