ISSN:0975-9646

A Review on Software Testing Framework in Cloud Computing

D.Anitha[1] and Dr.M.V.Srinath[2]

[1] Research Scholar, Department Of Computer Science, STET Womens College Mannargudi.

[2]Director, Department of Computer Applications, STET Womens College, Mannargudi.

Abstract— **Cloud computing has emerged as a new computing paradigm that impacts several different research fields, including software testing. In cloud computing, the user can use high end services in form of software that resides on different server and can be accessed from all over the world. It not only changes the way of obtaining computing resources but also alters the way of managing and delivering computing services, technologies and solutions. Software testing reduces the need for hardware and software resources and offers a flexible and efficient cloud platform. Testing in the cloud platform is effectively supported by engineers based on new test models and criteria. Prioritization technique is introduced to provide better relationship between test cases. These test cases are clustered based on priority level. The resources are utilized well by implementing load balancing algorithm. Cloud ensures optimal usage of available resources. However, security and privacy concerns are considered as a main obstacle in cloud. This paper surveys various prioritization, clustering, load balancing and security techniques to enhance the cloud environment. Moreover, the comparison between various software testing techniques are demonstrated.**

Index Terms—**Cloud computing, Cluster, Prioritization, Privacy, Security, Test cases.**

I. INTRODUCTION

Cloud computing has seized a substantial attention recently as it changes the way of computation and services to customers. It is a computing prototype, where a large pool of systems are connected in private or public networks. Cloud computing also provides dynamically scalable infrastructure for application, data and file storage. Cloud services permits individuals and businesses to use software and hardware that are managed by third parties at remote locations. Examples of cloud services includes,

- Online storage
- Social networking sites
- Webmail
- Online business applications

In presence of network connection, the information and computer resources can be accessed from anywhere. It contains a shared pool of resources, including data storage space, networks, computer processing power, specialized corporate and user applications. Cloud providers offer services that are clustered into three categories namely,

1. Software as a Service (SaaS)-Highly scalable internet based applications, which are introduced on the cloud.
2. Platform as a Service (Paas) - It contains considerable potential to help enterprise developers. It is based on .NET or Java extended with cloud services.
3. Infrastructure as a Service (Iaas) - It delivers cloud computing infrastructures, namely, servers, software, data-center space and network requirements.

The growth of cloud based services is evident, but cloud still undergoes development and testing before deploying. The development of cloud-based services increases with the need for testing their applications. Cloud computing provides cost effective and flexible means via scalable computing power and diverse services. The cloud computing deployments models are as follows,

1. Private cloud
2. Public cloud
3. Hybrid cloud
4. Community cloud

The third party uses the private cloud and these clouds are managed by the cloud computing provider. In the public cloud, the cloud infrastructure is made available to the general public or owned by the cloud providers. Hybrid cloud is a combination of two or more general cloud. The cloud shared by several organization are called as community cloud. Cloud computing solution depends on the availability of the infrastructure and the necessary business applications for their customers.

The features of cloud computing involves on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. In On-demand self-service, the customer request and manages their own computing resources. Broad network access permit services to be issued over the internet. The customer can use resources from a pool of computing resources. The need of hardware and software resources are minimized by the software testing technique in cloud. Cloud based software testing refers to testing and measurement activities on a cloud based environments. It also offers a flexible and efficient alternative to the traditional software testing process. It has low entry barriers and reduces cost by leveraging with computing resources in cloud. The cloud based testing ensures the quality of the cloud-based applications deployed in a cloud. It also checks for the automatic functional services. Testing a cloud refers to the verification and validation of applications, environments and infrastructure.

Moreover, software testing techniques are applied to the cloud environment in order to improve the security level. Test cases are scheduled and prioritized by prioritization techniques in order to maximize the scope.

Software test engineer schedule the test cases in a sequential order to achieve the code coverage at the fastest rate. In general, the time needed to run all the test cases in the test suite is long. The profits obtained through test case prioritization method becomes more significant. Clustering models are useful to software service providers to access their own services to the cloud users. It also helps the service provider to increase the availability of software service on the cloud environment. It allows cloud users to estimate potential software services available on the cloud computing environment. Load balancing in cloud environment provides an efficient solution to various issues residing in cloud set-up and usage. It considers two tasks, namely resource allocation and task scheduling. It also ensures that the resources are available based on the demand. And the resources are effectively used under the condition of high/low load. This paper presents various techniques and algorithms that are used to formulate software testing in cloud platform.

This paper is organized as follows. Section 2 describes the various test case prioritization, clustering, load balancing and cloud security techniques. Section 3 deals with results and discussion. Section 4 provides conclusions.

II. Testing Platform in Cloud Computing

The testing framework in cloud computing involves four major methods, namely,

1. Test case prioritization
2. Clustering techniques
3. Load balancing scenarios
4. Security Mechanism

A. PRIORITIZATION TECHNIQUES

Prioritization and test case scheduling is done in order to increase the objective of the function. These techniques provides a way to schedule and run test cases, which results in predicting the faults earlier. Some of the prioritization techniques are Dependency Structure Prioritization (DSP), Requirement based prioritization, Coverage based prioritization, Cost effective based prioritization and Chronographic based prioritization.

1. Dependency Structure Prioritization (DSP)

The dependency structure between test cases is closely related to the interaction between the parts of the system [1]. The priority for the test cases is assigned based on graph coverage value. The graph coverage value of a test case can be defined as the measurement of the complexities of the dependent test cases. The graph coverage value of a test case is measured based on the following ways,

1. The total number of dependents of the test case
2. The longest path of direct and indirect dependents of the test case.

Using these values, the depth-first search algorithm is used to calculate the priority of tests. For closed dependency structures, three ways are used to measure the graph coverage value. The three coverage measures for paths are

1. The number of non-executed test cases in the path.
2. The ratio of non-executed test cases in the path, with a higher weighted test cases towards the end of the path.
3. The number of non-executed test cases divided by the height of the path.

The DSP sum coverage measure reveals a higher weighted path comprising of more non-executed test cases. It finds a balance between longer paths, with few non-executed test cases. The DSP sum of a path p is defined as follows

$$DSP\_sum(p) = \#\{i \in 1\#p | \neg seen(t_i)\} \quad (1)$$

In which the # operator returns the size of a list or set. The DSP ratio coverage denotes the higher weight to paths that have higher ratio of non-executed tests. DSP ratio is calculated by using the below formula

$$DSP_{ratio}(p) = \frac{\sum_1^{\#p} w(t_i)}{\#p}, \quad (2)$$

$$w(t_i) = \begin{cases} i, if \ \neg seen(t_i), \\ 0, otherwise \end{cases} \quad (3)$$

The weighted sum of the path is calculated, which gives the weight of a test case and index of the path [2]. The worst-case complexity for calculating the DSP ordering is computed from the following equation.

$$O\big(|V| + |E| + 2^{|2V|}/2\big). |V| + |E| \quad (4)$$

The time required to calculate as a set of linearly independent paths in the graph is less. Fault rate detection is high and decreases the time consumption.

2. Requirement-Based Prioritization Techniques

The weight factors used in the prioritization techniques are customer priority, requirement complexity and requirement volatility [3]. The higher factor values denotes a need for prioritization of test case related to the requirement. It adjusts the coverage information for remaining test cases and recursively selects a test case that yields the greatest coverage of requirement. The test cases with highest Weight Prioritization WP are executed first. The weight priority is calculated from the following formula

$$P = (\sum i \ wiWi) \,|(\sum i \ Wi) \quad (5)$$

The factors that affect requirement based prioritization are time-to-market limitations, number of stakeholders, implementation cost. It does not consider the dependency of test cases.

3. Coverage-Based Prioritization Techniques

Test coverage analysis is a degree used in software testing known as code coverage analysis [4]. It deals with the amount of source code of program that has been exercised during the testing process. It is a form of white box testing, which checks the code directly. The process involved in coverage-based techniques are described as follows,

1. Discovering the areas of a program that are not exercised by a set of test cases.
2. Creation of additional test cases to increase coverage
3. Determining a quantitative measure of code
4. Identification of redundant test cases

It is a structural white box testing, which compares the test program behavior with the apparent intention of the source code. The coverage prioritization technique prioritize the test cases in decreasing order. The weight of the test cases in decreasing order are determined by the following equation:

$$TW = ReqSlice + ReqExercise \qquad (6)$$

TW denotes the weight prioritization computed for each test case. ReqSlice indicates the number of requirements depicted in the relevant slice of output for each test case. ReqExercise is a number of requirements exercised by test case. This technique contains the extra comment line, which is useless and the fault tolerance is low. It has high code complexity that means the dependency of the test case on other test case.

4. Cost Effective-Based Prioritization Techniques

In this technique, the prioritization of test cases are carried out based on cost analysis [5]. The actual time is measured for estimating the cost for each test cases. The following variables are used to prioritize the test cases where cost of analysis Ca(T) and cost of the prioritization algorithm, Cp(T).

$$WP = Ca(T) + Cp(T) \qquad (7)$$

WP is a weight prioritization value for each test case. Ca(T) includes the cost of source code analysis, Cp(T) is the actual cost for running a prioritization tool and relies on the algorithm used. It concentrates only on the cost factor and does not focus on risk factors.

5. Chronographic history-based Prioritization Techniques

The test cases are prioritized based on the test execution history [6]. By using historical information of the test cases and fault severities of the defects, which are covered by the test cases in a test suite. The historical value of the test cases is calculated and is used for the basis of the prioritization of the test cases. Historical value is calculated from the previous cost and fault severity of the defect, which are covered by a test case in suite. Weight factor for the history based prioritization is defined as the

$$wC_i = \frac{\overline{C_l}}{\overline{C_l} + \overline{FS_l}} \qquad (8)$$

Where $\overline{C_l}$ is the mean value of C, relative cost and $\overline{FS_l}$ is the mean value of FS, total relative fault severity. The fault severity is high and consumes more time.

B. CLUSTERING TECHNIQUES

1. Agglomerative Clustering

Agglomerative hierarchical clustering is a bottom-up clustering method, where the clusters consists of sub clusters [7]. It is used in many areas because of its ability to use arbitrary clustering dissimilarity or distance function. It is a greedy algorithm that considers a set of points, which integrates geometric and non-geometric properties. A binary clustering tree is constructed along with a cluster dissimilarity function. The data points are initially considered as clusters of size 1. At, each step, it selects the best pair of clusters that are not part of a larger clusters and integrates them into a single larger cluster. The process repeats until a single cluster containing all the data points is created.

```
$agglomerative\ (S = \{x_1, \dots\dots, x_n\})$
$returns\ Dendrogram_k for\ k = 1\ to\ |S|$
$C_i = \{x_i\}\forall_i,$
$for\ k = |S| down\ to\ 1$
$Dendrogram_k = \{C_1, \dots\dots, C_k\}$
$d(i,j) = D(C_i, C_j), \forall_i, j; l, m = argmin_{a,b} d(a,b)$
$C_l = Join(C_l, C_m); Remove\ (C_m)$
$endloop$
```

The cluster dissimilarity function d (A, B) also called as the distance function, computes the dissimilarity between the two clusters. The clusters with smallest value are chosen and grouped them into a single large cluster [8]. In agglomerative hierarchical clustering the distance are measured using the following formulas.

- $d(C_i, C_j) = min_{x \in C_i, x' \in C_j} d(x, x')$ called as single linkage, which produces long and skinny clusters. The clustering can be stopped by setting the threshold value using the equation.

- $d(C_i, C_j) = max_{x \in C_i, x' \in C_j} d(x, x')$ termed as complete linkage and the clusters are compact and equal in diameter.

- $d(C_i, C_j) = \frac{\sum x \in C_i, x' \in C_j d(x, x')}{|C_i| . |C_j|}$ is used to compute the distance between two items.

Each agglomeration takes place at a greater distance between clusters. In this, the smaller clusters are generated, which are useful for discovery. It produces an ordering of the objects, which are informative for data display. It has high flexibility regarding the level of granularity. It can be applicable to any attribute type and can easily handle any form of similarity or distance.

2. Cosine Similarity Algorithm

The similarity based clustering is used for grouping the similar test cases in order to make the testing effective. Cosine similarity approach is used to form the clusters [9]. In the distributed system clusters are established, which provides better performance in fault detection. The number of clusters is less than the number of distributed environment in order to form the group of the clusters. It is a measure of similarity between two vectors of an inner product space. This algorithm measures the cosine of the angle between the vectors.

$$sim(d_i, d_j) = cos(d_i, d_j) = d_i^t d_j \qquad (9)$$

The similarity of two document vectors $d_i, d_j, sim(d_i, d_j)$ is defined as the cosine of the angle between the vectors. The inner product is calculated from the above equation for unit vectors. Cosine measure is used in a variant of k-means called spherical means [10]. The k-means aims to reduce the Euclidean distance and spherical k-means maximizes the cosine similarity between documents in a cluster and the cluster's centroid is measured by:

$$max \sum_{r=1}^{k} \sum_{d_i \in s_r} \frac{d_i^t c_r}{\|c_r\|} \qquad (10)$$

Cosine similarity is particularly used in positive space, where the outcome is bounded in [0, 1]

$$similarity = \cos(\theta) = \frac{A.B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n}(A_i)^2 * \sqrt{\sum_{i=1}^{n}(B_i)^2}}} \quad (11)$$

The accuracy of this algorithm is low and the computation cost is high.

3. K-means Clustering

In K-means clustering, the clusters are formed from a set of objects based upon the squared-error objective functions:

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} |p - m_i|^2 \quad (12)$$

In the above expression, c_i are the clusters, p is a point in a cluster c_i and m_i the mean of cluster [11]. The mean of the cluster is denoted by a vector, which contains for each attribute. The input parameter for the mean values of the data objects is the number of clusters. It primarily takes the number of components of the population equal to the required number of clusters. The required number of clusters are selected where the points are mutually farthest apart. It identifies each component in the population and allocates it to one of the cluster that relies on the maximum distance. The position of centroid is recalculated every time and a component is added to the cluster. This process continues until the components are grouped into the final required number of clusters. K-means algorithm is highly sensitive to initial conditions. The issues arises when clusters are differing from sizes, densities, and non-globular shapes.

4. Partition-based Clustering

This algorithm is built based on the partition of the data, where each cluster optimizes a clustering criteria [12]. It also reduces the sum of squared distance from the mean within each cluster. This algorithm, reduces the clustering criteria by iteratively relocating the data points between the clusters until an optimal partition is obtained. The number of different partition for n observation into k groups is represented by the following equation,

$$S_n(k) = \frac{1}{k!} \sum_{i=0}^{i=k} (-1)^{k-i} \binom{k}{i} i^n \quad (13)$$

The complexity of partition clustering is large because it enumerates all possible groupings and tries to find global optimum.

C. LOAD DISTRIBUTION SCENARIOS

Load balancing ensures resource utilization by provisioning of resources to cloud users on demand basis. It supports prioritizing users by applying appropriate scheduling criteria.

1. Hierarchical Load Balancing

Hierarchical load balancing involves different levels of the cloud in load balancing decision [13]. This technique mostly operates in master slave mode. The tree data structure is used to model the hierarchical load balancing. Every node in the tree is balanced under the supervision of the parent node. The light weight agent process is used by master to obtain the statistics of the slave nodes or child nodes. Scheduling decision is made based on the information gathered by the parent node. Request monitor acts as a head of the network and is responsible for monitoring the service, which in turn monitor service nodes. Nodes at different levels of hierarchy communicate with the nodes below them to get information about the network performance.

In hierarchical load balancing, nodes are clustered into m regions. The nodes within the same region are known as local nodes. Nodes that have link to another regions are called as gateways. The regions are called as neighbors, when there is one or two links between nodes in the two regions. Each node in the network are uniquely identified by two identifiers called i and j. Identifier (i) denotes the region to which the node belongs and identifier j indicates the nodes present in the region i. Every node in the hierarchical load balancing comprises of routing table [14]. The routing table of each process p [i, j] consists of two small tables named prs (proximity route selection) and rgn (routing group name). The rgn table determines the preferred neighbor for a destination process, whose region is other than i. The prs table identifies the preferred neighbor for a destination process whose region is i.

- prs is a local routing table with n entries, where n is the maximum number of nodes in the local region. Each entry j denotes the cost of the path to the local node j and a preferred neighbor node along the path.
- rgn is a global routing table with m entries, in which m is the maximum number of regions in the network. Each entry contains the cost of the path to a destination region and a preferred neighbor node along the path to the destination region.

When a job request comes, the scheduler initializes job parameters and finds the Expected computing power, ECP for each job using the below equation

$$ECP_i = \sum_{k=1}^{t} CPUspeed_k / t \quad (14)$$

It is useful in medium or large size network with heterogeneous environment.

2. Static Load Balancing

In static environment, the cloud provider installs homogeneous resources [15]. When the environment is made static, the resources in the cloud are not flexible. In this scenario, the cloud requires prior knowledge of nodes capacity, processing power, memory performance and statistics of user requirements. The user requirements are not subjected to any change during the run-time. Static load balancing algorithms defines the task to the node based on the ability of the node to process new requests. The process is based on prior knowledge of the nodes' properties and capabilities. It also includes nodes' processing power, memory and storage capacity. Round robin algorithm provide a load balancing in static environment. It is not highly flexible and not scalable.

3. Dynamic Load Balancing

In dynamic environment the cloud provider installs the heterogeneous resources [16]. In dynamic environment the resources are flexible and does not rely on the prior knowledge by considering run-time statistics. The dynamic load balancing algorithms includes different attributes of nodes' capabilities and network bandwidth. The algorithm is a combination of prior gathered information about the nodes in the cloud. Based on the attribute gathered, the algorithm assigns and reassigns the task to the nodes dynamically. In the distributed system, the load decision is made based on computation of current load in every node. It selects the node with the minimum value of counter variable to the nodes and assign the application request to the selected node. This type of load balancing is complex and time consuming.

4. Centralized Load Balancing

In this technique, the allocation and scheduling decisions are made by a single node [17]. This node is responsible for storing the knowledge based on entire cloud network. It can be applied to static or dynamic approach for load balancing. The server or single node maintains the statistics of the entire network and updation takes place regularly. Centralized load balancing needs few message to achieve load balancing within system. It comprises of one centralized node CS, which acknowledges the incoming jobs. Each job has its arrival time and processing requirements associated with it. It is highly overloaded and has low fault tolerant capacity.

5. Distributed Load Balancing

In distributed load balancing, no single node is responsible for making resource provisioning or task scheduling decision [18]. The multiple domain monitors the network to make accurate load balancing decision. In static environment, every node in the network maintains the local knowledge to assure an efficient distribution of tasks whereas the re-distribution takes place in dynamic environment. Honey bee foraging is a self-organizing algorithm, which data structure for the implementation. In the network, all the processors store the own local database. The algorithm used in distributed load balancing are highly complex. The communication overhead is high in the distributed load balancing.

D. SECURITY MECHANISM

1. Diffie-Hellman Algorithm

This algorithm enables each party to generate a shared secret key for encryption and decryption of data [19]. It removes the need of transferring keys between two communicating parties. It arranges all group members in a logic ring or a binary tree and to exchange DH public keys. The random parameters generates a new shared keys for each message that is exchanged between sender and receiver. It is a cryptographic protocol, which establishes a shared secret key over an insecure communication channel by permitting the two parties that have no prior knowledge of each other [20]. This key is used to encrypt subsequent communications using a symmetric key cipher. The following steps describes a DH exchange

Step 1: Alice and Bob agree on generator g and module p.

Step 2: Alice selects a random large integer X_A and sends Bob its public value $Y_{A,}$ where $Y_A=g^{x(A)}$ mod p.

Step 3: Bob selects a random large integer X_B and sends Alice his public value $Y_{B,}$ where $Y_B=g^{x(B)}$ mod p.

Step 4: Alice computes k= $Y_B^{x(A)}$ mod p.

Step 5: Bob computes k'= $Y_A^{x(B)}$ mod p.

Step 6: Both k and k' are equal to $g^{x(A)\,x(B)}$ mod p.

It is used in interactive transaction rather than a batch transfer from a sender to receiver. It is used in many protocols, namely Secure Socket Layer, Secure Shell, Internet Protocol Security and Public Key Infrastructure.

2. Wang's approach

This approach deals with the security issues in cloud [21]. It provides data access procedure, which is based on owner- write user read scenario. In this approach end user sends a request to access the data to the data owner then the data owner sends an encryption key and access certificate to user, user sends the access certificate to storage provider and the storage provider sends the encrypted detail to the end user. It requires support from the cloud side and no multiple policies combination are used.

3. Data Encryption Standard (DES) algorithm

DES structure has a 64-bit block size and uses a 56 bit key during execution [22]. The key actually looks like a 64 bit quantity, but one bit in each of the octets is used for odd parity on each octet. For decryption the same algorithm is used, but the order of sub keys is reversed. It can process with an initial permutation, 16 rounds block cipher and final permutation. The application of the DES algorithm is very widespread in military, commercial, and other domains. Even though, this algorithm is public and the design issues used are classified. It has some drawbacks particularly in the selection of 56 bit key algorithm as it can be vulnerable to brute force attacks. In order to improve this, 2DES and 3DES algorithms are developed.

4. Advanced Encryption Standard (AES)

AES acts as a substitution-permutation network based on a design principle [23]. It operates on a 4x4 column-major order matrix of bytes and the matrix computations are carried out in a special finite field. The final output of cipher text is obtained by a number of repetitive transformation called as AES cipher. The number of cyclic repetition are described as follows,

- 10 cycles of repetition for 128 bit keys
- 12 cycles of repetition for 192 bit keys
- 14 cycles of repetition for 256 bit keys

Each round of encryption process requires the following four types of operations: SubBytes, ShiftRows, MixColumns, XorRoundKey. Decryption is the reverse process of encryption and using inverse functions: InvSubBytes, InvShiftRows, InvMixColumns. The major demerits of this algorithm is that it requires more round of communication as compared with shared key mechanism.

5. *Triple DES (3DES)*

The 3DES algorithm is essential for the replacement of DES algorithm due to its improvement on key searching [24]. It has three round message. It provides strongest encryption algorithm, since it is harder to break the 2^168 possible arrangements. It reduces the memory requirements among the keys. The major drawback of this algorithm is too time consuming.

TABLE 1 INFORMATION ABOUT DIFFERENT ENCRYPTION ALGORITHMS

Factors	AES	3DES	DES
Key Length	128, 192, or 256 bits	168 and 112 bits	56 bits
Cipher Type	Symmetric block cipher	Symmetric block cipher	Symmetric block cipher
Block size	128, 192, or 256 bits	64 bits	64 bits
Developed	2000	1978	1977
Security	Considered secure	Secured, but exit in DES	Proven inadequate

III. Results and Discussion

Various techniques for software testing in cloud platform are discussed. The results of the survey are shown in Table 2. Test case prioritization techniques effectively schedules, run the test cases and also predicts the fault earlier. From the survey, it is proven that the dependency structure prioritization reduces the fault at low cost than the existing techniques. The test cases are clustered effectively by agglomerative clustering, which is easy to handle and applicable to any type of attribute. Hierarchical Load balancing ensures better resource utilization than the other scenarios such as static, dynamic, centralized and distributed. Moreover, the surveyed results evidently shows that the security of the cloud is enhanced by using Diffie-Hellman algorithm.

TABLE 2 INFORMATION ABOUT SOFTWARE TESTING TECHNIQUES IN CLOUD PLATFORM

Techniques	Author & Reference	Year	Performance	Quality Measurement
Test Case Prioritization Techniques				
Dependency Structure Prioritization	Zhang, et al [25]	2014	Every dependency test is reported and based on the report test case is reordered.	1. Detected dependency test 2. Cost analysis
	Haidry and Miller [1]	2013	It uses dependency information from a test suite to prioritize that test suite.	1. Artifact 2. Testing type 3. Lines of code 4. Functions 5. Faults 6. Tests 7. Dependencies 8. Graph density 9. Unconnected tests 10. Maximum depth
Requirement-Based Prioritization Technique	Berander and Andrews [26]	2005	It considers different aspects, techniques and stakeholders situation.	1. Requirement 2. Cost 3. Time 4. Risk 5. volatility
	Lehtola, et al [27]	2004	It requires complex context-specific decision making and performed iteratively.	1. priority list of local areas 2. prioritization scales 3. Negotiation in meetings
Coverage-Based Prioritization Techniques	Mohanty, et al [4]	2011	It generates a set of tests in orders and pair-wise interactions are tested.	1. Weight covered in each row using unweighted and weighted density 2. Cumulative weight covered 3. Size of test suites 4. Percentage of pairs involved
	Bryce, et al [28]	2011	It reduces test suites by using tests that provide coverage of the requirement.	1. Order suite function 2. Weighted frequency 3. Best and worst case criteria for prioritization 4. Fault detection
Cost Effective-Based Prioritization Techniques	Malishevsky, et al [29]	2006	It permits practitioners to perform prioritization by considering the cost.	1. Test suite executed 2. Total test case cost incurred 3. Evaluating fault severities
Chronographic history-based Prioritization Techniques	Lin, et al [30]	2013	It considers both source code information and historical fault data.	1. Test case 2. Fault detection 3. Fault-prone test cases 4. Repeated fault detection
	Park, et al [6]	2008	It is based on the use of historical information to estimate the cost and fault severity	1. Relative cost 2. Total fault severities 3. Relative fault severity

Techniques	Author & Reference	Year	Performance	Quality Measurement
Clustering Algorithm				
Agglomerative Clustering	Shalom and Dash [32]	2014	It generates high level multiple partitions in a given dataset.	1. Half distance matrix computation 2. Computational time
	Srivastava, et al [31]	2013	It increases efficiency by executing the task in parallel and suitable for large data set.	1. Number of nodes 2. Clustering analysis
	Davidson and Ravi [8]	2005	It can cause the dendogram to stop prematurely in a dead-end solution.	1. Data set 2. Unconstrained 3. Constrained
Cosine Similarity Algorithm	Hayes and Avesani [33]	2007	The similarity based clustering is used for grouping the test cases in order to make testing effective.	1. Mean fraction of a cluster 2. Queries generated
	Basu, et al [34]	2004	It can be parameterized using a symmetric positive definite matrix.	1. Cluster selection 2. Pairwise supervision
K-means Clustering	Liu, et al [35]	2013	It initially takes the number of components of the population equal to the final required number of clusters.	1. Various clusters 2. Dataset sizes 3. Overhead at the server side
	Peters [36]	2006	It classifies the group of objects based on attributes or features into k numbers of group.	1. Initial cluster assignment 2. Number of objects in the boundary area
	Zhao, et al [37]	2009	In each iteration it needs a total number of distance computations.	1. Speedup 2. Scale up 3. Size up
Partition-based Clustering	Hu, et al [38]	2006	This algorithm is efficient for large class hierarchies and the time complexity is O(n^2)	1. Labeling blocks 2. Partitioning 3. Matching block
	Salzbrunn [39]	2008	It partitions the whole domain according to certain characteristics which are based on vector values.	1. Hedgehog representation of the flow.
Load Balancing Scenarios				
Hierarchical Load Balancing	Han and Chronopoulos	2013	It reduces the communication time by loop scheduling.	1. Total execution time for quick sort 2. Total execution for matrix multiplication
	Zheng, et al [13]	2010	It divides the processors into independent autonomous group and organize the group in a hierarchy manner	1. Memory usage 2. Computation of time
Static Load Balancing	Penmatsa and Chronopoulos [15]	2011	In this scenario, the cloud requires prior knowledge of nodes capacity, processing power, memory performance and statistics of user requirements.	1. Expected response time 2. Fairness index 3. System utilization
	Pan, et al [40]	2007	Static load balancing algorithms assign the task to the node based on the ability of the node to process new requests.	1. Speedup 2. Efficiency of thread
Dynamic Load Balancing	Dhakal, et al [41]	2007	These are executed by considering the average completion time per task and the processing rate in presence of external loads.	1. Number of tasks 2. Mean delay 3. The amount of load transferred between nodes.
	Chieu, et al [42]	2009	It allows system to automatically and dynamically add new web server instances.	1. Utilization rates 2. Low power usage cost 3. Average response time

Techniques	Author & Reference	Year	Performance	Quality Measurement
Centralized Load Balancing	Saxena and sharma [43]	2011	In this, single node or server is responsible for maintaining the statistics of entire network and updating it from time to time.	1. Load matrix 2. Capability matrix
	Azzoni and Down [44]	2009	Centralized load balancing needs few message to achieve load balancing within system.	1. Average task completion time 2. Number of message exchange.
Distributed Load Balancing	Berebbrink, et al [45]	2007	In this, no single node is responsible for making resource provisioning or task scheduling decision.	1. Upper bounds on convergence time 2. Lower bound
	Menon and Kale [46]	2013	The multiple domain monitors the network to make accurate load balancing decision.	1. Load of under loaded processors 2. Probabilities assigned to each of the processors 3. Transfers received
Security Mechanism in Cloud Environment				
Diffie-Hellman algorithm	Tirthani and Ganesan [20]	2014	It agrees on a key that two parties can use for a symmetric encryption, in such a way that an attacker obtain the key.	1. Computation of point on the curve 2. Integer factorization 3. Key generation 4. Decryption
	Liu, et al [47]	2013	This information is used to build and design secure and efficient versions of the classic key agreement protocol.	1. Server Instances involved 2. Data Block Size 3. Encryption time 4. Exchange time 5. Key Exchange
Data Encryption Standard (DES) algorithm	Liao and Chao [48]	2008	It diffuses the encipherment transformation over the whole 64-bit cipher text within the 16 substitution and transposition rounds	1. Successful rate under toleration distance
	Benabdellah, et al [49]	2007	It can process with an initial permutation, 16 rounds block cipher and final permutation	1. Computation time 2. Compression ratio
Advanced Encryption Standard (AES)	Trang and Loi [50]	2012	It operates on a 128-bit block of data and executed Nr-1 loop times.	1. Timing simulation
	Yen and Wu [51]	2006	The data procedure is the main body of the encryption and consists of four operations.	1. Percentage of undetectable errors. 2. Error detection capability
Triple DES (3DES)	Ghosal, et al [52]	2010	It is a block cipher operating on 64-bit data blocks.	1. Logic utilization 2. Number of slice registers
	Antonios, et al [53]	2006	It increases the length of the used key since two or three keys are applied depending on which mode of operation is used.	1. Number of slices 2. Minimum period

IV. Proposed Work

The proposed testing framework in cloud computing will use Dependency Structure priority to prioritize the test cases and these prioritized test cases are clustered using Agglomerative clustering technique. Hierarchical load balancing mechanism will be applied to utilize the resources. In order to provide a secured cloud environment Diffie-Hellman algorithm will be applied.

V. Conclusion

In this paper, an overview of software techniques in cloud computing platform are depicted. From the survey, it proves that the Dependency Structure Prioritization (DSP) are very powerful to prioritize the test cases. It minimizes the time consumption and increases the fault rate prediction. The test cases are clustered using the agglomerative clustering, which provides high flexibility in the level of granularity. The resources are well-utilized based on Hierarchical load balancing algorithm. It can be applied in medium or large size cloud environment. The security of the cloud is enhanced by the Diffie-Hellman algorithm. An enhanced testing framework in cloud platform can be achieved by using the DSP, agglomerative clustering, Hierarchical load balancing and Diffie-Hellman techniques.

References

[1] S. Haidry and T. Miller, "Using dependency structures for prioritization of functional test suites," *Software Engineering, IEEE Transactions on*, vol. 39, pp. 258-275, 2013.

[2] Gurdiksha and J. Singh, "Approaches used for prioritization of Test Suites," *IJSER*, vol. 2, 2014.

[3] P. Berander and A. Andrews, "Requirements prioritization," in *Engineering and managing software requirements*, ed: Springer, 2005, pp. 69-94.

[4] S. Mohanty, A. A. Acharya, and D. P. Mohapatra, "A survey on model based test case prioritization," *International Journal of Computer Science and Information Technologies,* vol. 2, pp. 1042-1047, 2011.

[5] Karambir and R. Rani, "Prioritize Test Case Survey for Component-Based Software Testing," *ijarcsse,* vol. 3, 2013.

[6] H. Park, H. Ryu, and J. Baik, "Historical value-based approach for cost-cognizant test case prioritization to improve the effectiveness of regression testing," in *Secure System Integration and Reliability Improvement, 2008. SSIRI'08. Second International Conference on*, 2008, pp. 39-46.

[7] R. G. Pensa, D. Ienco, and R. Meo, "Hierarchical co-clustering: off-line and incremental approaches," *Data Mining and Knowledge Discovery,* vol. 28, pp. 31-64, 2014.

[8] I. Davidson and S. Ravi, "Agglomerative hierarchical clustering with constraints: Theoretical and empirical results," in *Knowledge Discovery in Databases: PKDD 2005*, ed: Springer, 2005, pp. 59-70.

[9] C. S. Rao and P. S. P. Rao, "A Novel Cosine Similarity Based Clustering Mechanism," 2012.

[10] G. S. Reddy and R. Krishnaiah, "Clustering algorithm with a novel similarity measure," *IOSR Journal of Computer Engineering (IOSRJCE). v4 i6,* pp. 37-42, 2012.

[11] K. A. Nazeer and M. Sebastian, "Improving the Accuracy and Efficiency of the k-means Clustering Algorithm," in *Proceedings of the World Congress on Engineering*, 2009, pp. 1-3.

[12] A. K. Upadhyay and A. Misra, "Prioritizing Test Suites Using Clustering Approach in Software Testing."

[13] G. Zheng, E. Meneses, A. Bhatele, and L. V. Kale, "Hierarchical load balancing for Charm++ applications on large supercomputers," in *Parallel Processing Workshops (ICPPW), 2010 39th International Conference on*, 2010, pp. 436-444.

[14] M. Katyal and A. Mishra, "A Comparative Study of Load Balancing Algorithms in Cloud Computing Environment," *International Journal of Distributed and Cloud Computing,* vol. 1, pp. 5-14, 2013.

[15] S. Penmatsa and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *Journal of Parallel and Distributed Computing,* vol. 71, pp. 537-555, 2011.

[16] N. Haryani and D. Jagli, "Dynamic Method for Load Balancing in Cloud Computing," *IOSR Journal of Computer Engineering (IOSRJCE). v4 i6,* vol. 16, pp. 23-28, 2014.

[17] K. A. Nuaimi, N. Mohamed, M. A. Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: Challenges and algorithms," in *Network Cloud Computing and Applications (NCCA), 2012 Second Symposium on*, 2012, pp. 137-142.

[18] M. Randles, D. Lamb, and Taleb-Bebdiab, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," *IEEE*, 2010.

[19] L. Harn and C. Lin, "Efficient group Diffie–Hellman key agreement protocols," *Computers & Electrical Engineering,* 2014.

[20] N. Tirthani and R. Ganesan, "Data Security in Cloud Architecture Based on Diffie Hellman and Elliptical Curve Cryptography," *IACR Cryptology ePrint Archive,* vol. 2014, p. 49, 2014.

[21] R. R. Sadul, N. Subhekar, A. Rankhambe, S. Shaikh, and Mokashi, "A Survey of Different Encryption Techniques for Secure Cloud Storage," *MJRET,* vol. 1, pp. 59-65, 2014.

[22] J. Thakur and N. Kumar, "DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis," *International journal of emerging technology and advanced engineering,* vol. 1, pp. 6-12, 2011.

[23] N. Singhal and J. Raina, "Comparative analysis of AES and RC4 algorithms for better utilization," *International Journal of Computer Trends and Technology,* vol. 79, pp. 177-181, 2011.

[24] F. Antonios, P. Nikolaos, M. Panagiotis, and A. Emmanouel, "Hardware Implementation of Triple-DES Encryption/Decryption Algorithm," in *International Conference on Telecommunications and Multimedia*, 2006.

[25] S. Zhang, D. Jalali, J. Wuttke, K. Muslu, W. Lam, M. Ernst, et al., "Empirically revisiting the test independence assumption," University of Washington Technical Report UW-CSE-14-01-01. Available at: http://homes. cs. washington. edu/~ szhang/pdf/testdependence. pdf, 2014.

[26] P. Berander and A. Andrews, "Requirements prioritization," in *Engineering and managing software requirements*, ed: Springer, 2005, pp. 69-94.

[27] Lehtola, M. Kauppinen, and S. Kujala, "Requirements prioritization challenges in practice," in *Product focused software process improvement,* ed: Springer, 2004, pp. 497-508

[28] R. C. Bryce, S. Sampath, and A. M. Memon, "Developing a single model and test prioritization strategies for event-driven software," *Software Engineering, IEEE Transactions on,* vol. 37, pp. 48-64, 2011.

[29] A. G. Malishevsky, J. R. Ruthruff, G. Rothermel, and S. Elbaum, "Cost-cognizant test case prioritization," *Department of Computer Science and Engineering, University of Nebraska-Lincoln, Techical Report,* 2006.

[30] C.-T. Lin, C.-D. Chen, C.-S. Tsai, and G. M. Kapfhammer, "History-based Test Case Prioritization with Software Version Awareness," in Engineering of Complex Computer Systems (ICECCS), 2013 18th International Conference on, 2013, pp. 171-172.

[31] K. Srivastava, R. Shah, D. Valia, and H. Swaminarayan, "Data Mining Using Hierarchical Agglomerative Clustering Algorithm in Distributed Cloud Computing Environment," *International Journal of Computer Theory & Engineering,* vol. 5, 2013.

[32] A. Shalom and M. Dash, "Parallel Computations for Hierarchical Agglomerative Clustering using CUDA," 2014.

[33] C. Hayes, "Using tags and clustering to identify topic-relevant blogs," 2007.

[34] S. Basu, M. Bilenko, and R. J. Mooney, "A probabilistic framework for semi-supervised clustering," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining,* 2004, pp. 59-68.

[35] R. Liu, P. Mordohai, W. H. Wang, H. Xiong, R. Liu, H. W. Wang, *et al.*, "Integrity Verification of K-means Clustering Outsourced to Infrastructure as a Service (IaaS) Providers," in *SDM*, 2013, pp. 632-640.

[36] . Peters, "Some refinements of rough k-means clustering," *Pattern Recognition,* vol. 39, pp. 1481-1491, 2006.

[37] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Cloud Computing*, ed: Springer, 2009, pp. 674-679.

[38] W. Hu, Y. Zhao, and Y. Qu, "Partition-based block matching of large class hierarchies," in *The Semantic Web–ASWC 2006*, ed: Springer, 2006, pp. 72-83.

[39] T. Salzbrunn, H. Jänicke, T. Wischgoll, and G. Scheuermann, "The State of the Art in Flow Visualization: Partition-Based Techniques," in *SimVis*, 2008, pp. 75-92.

[40] Y. Pan, W. Lu, Y. Zhang, and K. Chiu, "A static load-balancing scheme for parallel XML parsing on multicore CPUs," in *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, 2007, pp. 351-362.

[41] S. Dhakal, M. M. Hayat, J. E. Pezoa, C. Yang, and D. A. Bader, "Dynamic load balancing in distributed systems in the presence of delays: A regeneration-theory approach," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 18, pp. 485-497, 2007

[42] T. C. Chieu, A. Mohindra, A. A. Karve, and A. Segal, "Dynamic scaling of web applications in a virtualized cloud computing environment," in *e-Business Engineering, 2009. ICEBE'09. IEEE International Conference on*, 2009, pp. 281-286.

[43] A. B. Saxena and D. Sharma, "Analysis of threshold based centralized load balancing policy for heterogeneous machines," *International Journal of Advanced Information Technology (IJAIT),* vol. 1, 2011.

[44] I. Al-Azzoni and D. G. Down, "Decentralized load balancing for heterogeneous grids," in *Future Computing, Service*

Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:, 2009, pp. 545-550.

[45] P. Berenbrink, T. Friedetzky, L. A. Goldberg, P. W. Goldberg, Z. Hu, and R. Martin, "Distributed selfish load balancing," *SIAM Journal on Computing,* vol. 37, pp. 1163-1181, 2007.

[46] H. Menon and L. Kalé, "A distributed dynamic load balancer for iterative applications," in *Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2013, p. 15.

[47] C. Liu, X. Zhang, C. Yang, and J. Chen, "CCBKE—Session key negotiation for fast and secure scheduling of scientific applications in cloud computing," *Future Generation Computer Systems,* vol. 29, pp. 1300-1308, 2013.

[48] H.-C. Liao and Y.-H. Chao, "A new data encryption algorithm based on the location of mobile users," *Information Technology Journal,* vol. 7, pp. 63-69, 2008.

[49] M. Benabdellah, N. Zahid, F. Regragui, and E. H. Bouyakhf, "Encryption-Compression of Echographic images using FMT transform and DES algorithm," *INFOCOM International Journal, papier accepté en Mars,* 2007.

[50] T. Hoang, "An efficient FPGA implementation of the Advanced Encryption Standard algorithm," in *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on*, 2012, pp. 1-4.

[51] C.-H. Yen and B.-F. Wu, "Simple error detection methods for hardware implementation of advanced encryption standard," *Computers, IEEE Transactions on,* vol. 55, pp. 720-731, 2006.

[52] P. Ghosal, M. Biswas, and M. Biswas, "A Compact FPGA Implementation of Triple-DES Encryption System with IP Core Generation and On-Chip Verification," in *Proceding of the 2010 International Conference on Industrial Engineering and Operation Management, Dhaka, Bangladesh*, 2010.

[53] F. Antonios, P. Nikolaos, M. Panagiotis, and A. Emmanouel, "Hardware Implementation of Triple-DES Encryption/Decryption Algorithm," in *International Conference on Telecommunications and Multimedia*, 2006.