IMPROVISING RELIABILITY THROUGH N-VERSION PROGRAMMING IN CLOUD ENVIRONMENT

S Pramila[1], S Poonkuzhali[2], S Mythili[3]

[1]Professor, Dept of Information Technology, Rajalakshmi Institute of Technology, Chembarambakkam, Chennai, (India)

[2]Professor, [3]PG Scholar, Dept of Information Technology, Rajalakshmi Engineering College, Thandalam, Chennai, (India)

ABSTRACT

Cloud computing provides computing as a service rather than technology. The main advantage of computing includes agility, scalability and elasticity, low computational cost. Due to this the entrepreneurs are moving their application from the legacy environment to the cloud environment. But during the process of migration to the cloud environment, the reliability of the application is degraded. Thus a reliability based framework has been designed to improve the reliability. Since the failure rate of each component plays a vital role in determining the reliability, it becomes necessary to analyze each component separately. Incase of successive identification of these factors the component significance can be effectively predicted. Various fault tolerance strategies can be applied to improve the reliability.

Keywords: Cloud Migration, Fault Tolerance, Software Reliability

I. INTRODUCTION

Cloud Computing enables convenient, on-demand network access to a shared pool of configurable computing resources [1]. With the emergence of cloud computing and online services, customers expect services to be available whenever they need them just like electricity or dial tone. The characteristics of cloud include self-healing, multi tenancy, SLA driven, linearly scalable, virtualized and flexible. It provides automatic reconfiguration of resources based on the service level agreements.

Cloud computing involves running applications on virtual servers that are allocated on this distributed hardware infrastructure available in the cloud. These virtual servers are made in such a way that the different service level agreements and reliability issues are met. There may be multiple instances of the same virtual server accessing the different parts of the hardware infrastructure available. This is to make sure that there are multiple copies of the applications which are ready to take over on another one's failure. This expectation requires organizations that build and support cloud services to plan for probable failures and have mechanisms that allow rapid recovery from such failures.

Although much progress has already been made in cloud computing, there are a number of research areas that still need to be explored. Issues of security, reliability, and performance should be addressed to meet the specific requirements of different organizations, infrastructures, and functions [2]. A cloud should be able to continue to run in the presence of hardware and software faults. The application might require more stringent reliability that would be better served by a combination of more robust hardware and/or software-based fault-tolerance techniques.

Reliability has become the major concern as more users come to depend on the services offered by cloud. Reliability becomes increasingly important, especially for long-running or mission critical application. When an enterprise needs its Legacy applications to migrate to the cloud environment, it requires a deep understanding about the application to be migrated to cloud. FTCloud [3] which is a component ranking based framework needs expert knowledge to manually designate critical components and identifying the components significance during migration. An automatic selection strategy is proposed in ROCloud approach [4]. The various fault tolerance strategies like recovery block, n- version programming and parallel techniques are used to tolerate the faults during the migration of an application. It makes the system more robust instead of removing the fault components.

In this paper the significant components whose failure can have great impact on reliability of the application to be migrated is identified. The component graph gives the overall structure of the application. The fault tolerance strategies are applied based on the requirements of the enterprise.

II. RELATED WORKS

The following papers have specified the concept of improving the reliability of an application during migration.

Zibinzheng et al. [3] proposed a component ranking framework for fault tolerant application. It employs the component invocation structures and the invocation frequencies to identify the significant components in a cloud application. An algorithm is applied to automatically determine optimal fault tolerant strategy for these significant components. But it becomes insufficient approach due to the manual designating of the significant components.

WeiweiQiu et al. [4] proposed a reliability based design optimization for cloud migration. It includes two ranking algorithm one for migrating all the application to cloud and other for migrating hybrid application. Based on the ranking, it identifies the fault tolerance of each component and refactoring the small number of error prone components.

Doaa M. Shawky et al. [5] proposed a cost effective approach for hybrid migration to cloud. In this approach, coupling among different components of the system is measured. Then, a proposed cost measuring function is used to choose the optimal migration scenarios.

Kasto Inoue et al. [6] proposed a component rank: Relative significance rank for software component search. It analyzes the actual use relations among the components and propagating the significance through the use relations. It has considered only the source code as component while leaving other components for ranking.

III. METHODS AND MATERIALS

The methods and materials for the proposed system include the various modules. The Fig. 1 describes the architecture of the proposed system.

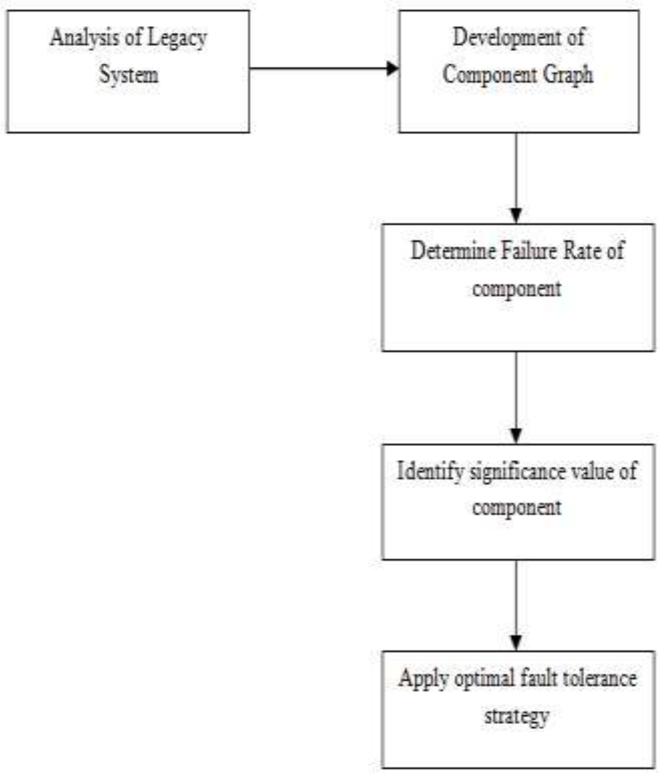
Analysis of Legacy System
Development of Component Graph
Determine Failure Rate of component
Identify significance value of component
Apply optimal fault tolerance strategy

Fig: Framework of the Proposed System.

3.1 Analysis

The legacy system to be migrated is analyzed. The result of the analysis provides the information about the various components of the application. The component graph depicts the invocation and coupling information among the components.

3.2 Development of Component Graph

3.2.1 Algorithm

Input: Component Information

Process:

1. Declare an array of M[size][size] which will store the graph.
2. Enter the nodes.
3. Enter the edges of the graph by two vertices say Vi, Vj
4. If the graph is directed set M[i][j]=1.
5. When all the nodes are connected through edges identify the invocation information.

Output: Adjacency Matrix.

3.2.2 Algorithm

Input: Component Information.

Process:

1. Define the dep_resolve of a node.
2. For each edge connected to the defined node, define dep_resolve.
3. Continue step2 for all the nodes in the graph.

4. Generate the dependency matrix

$$DM = \begin{matrix} d11 & d12 & \cdots & d1n \\ d21 & d22 & \cdots & d2n \\ \cdot & \cdot & \cdot & \cdot \\ dn1 & dn2 & \cdots & dnn \end{matrix}$$

Where,

$$dij = cdij + ddij + tdij + sdij + rdij + ioij + edij$$

Output: Dependency Matrix.

3.3 Determining Failure Rate of Component

The failure rate of the component plays a vital role in analyzing the significance of the component. The impact varies from one component to another. In case of higher the failure rate higher is its impact. It is calculated as number of times the component failed to the number of times the component is invoked.

The higher the failure rate, lower is its significance value. After analyzing these components the component can be ranked according to their significance value.

3.4 Fault Tolerance Technique

The N- version programming (NVP) strategy is used to tolerate the fault in the component application. In this approach the failure component is replaced by applying the NVP approach. The component has n- functionally independent and equivalent programs. The implementation of these components differ in algorithms and programming language. Based on the voting and comparison criteria the best component or the module to be replaced is selected.

3.5 Optimal Design

The above steps are applied to build a design that helps in enhancing the reliability of the application. For each component in the application, their significance and its failure rate provides a great knowledge to developer to design the system effectively

IV. CONCLUSION

The reliability of the system is the property to maintain constantly in an application. Thus by identifying the failure rate of each component the reliability of the application can be greatly improved. The application contains great influence to various factors like latency, response time and cost. The NVP approach helps in providing the replica of each module and to tolerate the failure to some extent. The future work can be extended to analyzing the other factors in determining the cost for feasible migration.

REFERENCE

[1] Mike adams, Shannon bearly ,David bills, an introduction to designing a reliable cloud services, January 2014.

[2] P.Melland T.Grance, the NIST definition of cloud computing Recommendations of the National Institute of Standards and Technology, NIST special publication, 2011.

[3] ZibinZheng,TomChao, Micheal R lyu, Irwin King, FTCloud- a componenet ranking framework for fault tolerant cloud application, International symposium on Software Reliability Engineering, 2010.

[4] Weiwei Qui, ZibinZheng, Xinyu Wang, Xiaohu Yang, Reliability based design optimization for cloud migration, IEEE transaction on service computing, vol 7,April-June 2014.

[5] Doaa M. Shawky, A Cost-effective Approach for Hybrid Migration to the Cloud, International Journal of Computer and Information Technology (ISSN: 2279 – 0764) Volume 02– Issue 01, January 2013.

[6] Katsuro Inoue, Component Rank: Relative Significance Rank for software component search.