www.ijecs.in
International Journal Of Engineering And Computer Science ISSN: 2319-7242
Volume 5 Issue 5 May 2016, Page No. 16643-16646

Optimized cloud migration using reliabilty framework

Ancy Nevil.S, Kavitha Priya . C.J

1 Jeppiaar Institute of Technology
Chennai,India
ancys@jeppiaarinstitute.org

[2]Jeppiaar Institute of Technology
Chennai,India
kavithapriyacj@jeppiaarinstitute.org

Abstract--- ***The on-demand use, high scalability, and low maintenance cost nature of cloud computing have attracted more and more enterprises to migrate their legacy applications to the cloud environment. Although the cloud platform itself promises high reliability, ensuring high quality of service is still one of the major concerns, since the enterprise applications are usually complicated and consist of a large number of distributed components. Thus, improving the reliability of an application during cloud migration is a challenging and critical research problem. To address this problem, we propose a reliability-based optimization framework, named RO Cloud, to improve the application reliability by fault tolerance. RO Cloud includes two ranking algorithms. The first algorithm ranks components for the applications that all their components will be migrated to the cloud. The second algorithm ranks components for hybrid applications that only part of their components are migrated to the cloud. Based on the ranking result, optimal fault-tolerant strategy will be selected automatically for the most significant components with respect to their predefined constraints. The experimental results show that by refactoring a small number of error-prone components and tolerating faults of the most significant components, the reliability of the application can be greatly improved.***

Keywords—RO Cloud, Refactoring, Information Flow Control(IFC)

I. Introduction

Cloud computing enables convenient, on-demand network access to a shared pool of configurable computing resources.In the cloud computing environment, the computing resources can be provisioned to users on-demand, like the electricity grid [21]. Startup companies can deploy their newly developed Internet services to the cloud without the concern of upfront capital or operator expense [3]. However, cloud computing is not only for startups, its cost effective, high scalability and high reliability features also attracted enterprises to migrate their legacy applications to the cloud. Before the migration, enterprises usually have the concern to keep or improve the application reliability in the cloud environment [26] [4]. Thus, reliability based optimization when migrating legacy applications to the cloud environment is becoming an urgently required research problem [9]. In traditional software reliability engineering, there are four major approaches to improve system reliability: fault prevention, fault removal, fault tolerance, and fault forecasting. When turning to the cloud environment, since the applications deployed in the cloud are usually removal techniques are not sufficient [23]. Another approach for building reliable systems is software fault tolerance, which is to employ functionally equivalent components to tolerate faults.

Although the cloud platform is flexible and can provide resources on-demand, there is still a charge for using the cloud components [19]. At the same time, legacy applications usually involve a large number of components, so it will be expensive to provide redundancies for each component [15]. To reduce the cost so as to assure highly reliability in a limited budget during the migration of legacy applications to cloud, an efficient reliability-based optimization framework is needed.

II. Related work

Legacy applications usually involve a large number of components, so it will be expensive to provide redundancies for each component [10]. To reduce the cost so as to assure highly reliability in a limited budget during the migration of legacy applications to cloud, an efficient reliability-based optimization framework is needed. In the previous work, FT Cloud is proposed to improve the reliability of newly developed cloud applications, which identifies the most significant components depending on the structure information and expert knowledge of critical components[25]. Compared with newly developed applications, the reliability-based optimization of legacy applications has the following difficulties:

The failure rate of different components in a legacy application can vary [29]. For example, some components in the legacy application are implemented by out-dated technology and have not been well maintained. These components can have great impact on application reliability. But they may not be selected as significant component by FT Cloud, since FT Cloud only

employs structure information and does not take component failure rate information into consideration [20] [14].

FT Cloud needs expert knowledge to manually designate critical components. However, the migration team may not be the creator of the legacy application [1]. So it will be difficult for them to manually list the critical components. Furthermore, the number of legacy applications as well as the number of components in these applications is large, it is thus impractical to manually identify critical components [6].

Some applications may be restricted by enterprise security polices and only part of their components can be migrated to the cloud. The component ranking and fault-tolerant strategy selection algorithms should take these hybrid applications into consideration [30].

For these two reasons, FTCloud is not sufficient for improving the reliability of legacy applications. We need to take advantage of all materials of the legacy applications at hand, such as application logs, source code, etc. to automatically identify the components whose failures have great impact on the application reliability [11]. Then provide backups for them using redundant resources in the cloud to improve the application reliability [17]. Based on this idea, we proposed Reliability-based Optimization in Cloud environment (ROCloud), which is a component ranking framework based on historical information to identify the significant components that have great impact on application reliability, and suggest optimal fault tolerance strategies automatically [8]. ROCloud can help the designer optimize legacy application design to get a more reliable and robust cloud application effectively and efficiently [24].

III. System model

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development [22]. There is some overlap with the disciplines of systems analysis and systems engineering.

1) Authentication And Authorization

Authentication and Authorization process are the required processes to Verifying the User Originality and Appropriate Session Activities of the Registered User. For strong authorization use biometric application.

2) Optimization Framework

Reliability optimization framework, which includes three phases:

i) Legacy application analysis,

ii) Automated significance ranking,

iii) Fault tolerance strategy selection.

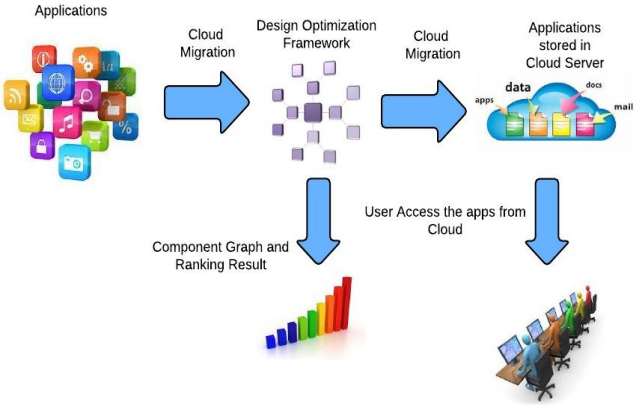


Fig 1: Framework for system design

The processes of each phase are as follows:

Both structure and failure information is extracted during the legacy application analysis phase. The structure information extraction consists of two sub processes: component extraction and invocation extraction. The failure information including failure rate and failure impact are collected from the execution logs and test results of the legacy application. Components with a failure rate higher than the threshold will be re-factored, and their reliability properties will be updated. A component graph is built for the legacy application based on the structure as well as the failure information. In the automated significance ranking phase, two algorithms are proposed for ordinary applications that can be migrated to public cloud and hybrid applications that need to be migrated to hybrid cloud, respectively [12].The performance, overhead, and cost of various fault tolerance strategy candidates are analyzed and the most suitable fault tolerance strategy is selected for each significant component based on its predefined constraint.

a. Legacy Application Analysis

The structure information includes components and the invocation information. The components are extracted from legacy applications by source code and documentation analysis [7]. The invocation information such as invocation links and invocation frequencies can be identified from application trace logs. Source codes and documentations are useful supplementary materials in addition to trace logs. All the information are represented in a component graph [16]. The main optimization goal is reliability, so a more straightforward way is employed to determine which components should be re-implemented: components with failure rates greater than a threshold [28]. The selection of the threshold is dependent on project budget and the target application failure rate, since extra development and testing effort is required for component reimplementation. After refactoring, the component failure rates will be estimated based on test results, and the component reliability property dataset will be updated.

b. Automated Significance Ranking

Based on the component graph, two component ranking algorithms are proposed in this section. The first algorithm ranks components for ordinary applications where all their components can be migrated to the cloud. The second algorithm rank components for hybrid applications which can be partly moved to the cloud [13]. In a distributed application,

the failures of the components which are frequently invoked by many other components tend to have greater impact on the system compared with the components which are rarely invoked by others.

Thus these components are considered to be more important from the reliability aspect and should be ranked at the front of component list [5]. Inspired by the Page Rank algorithm, we propose an algorithm to calculate the significance value of each component of the migratory application employing the component invocation relationships and reliability properties. Based on the component graph and component reliability information, the component ranking algorithm includes the following steps:

1) Initialize by randomly assigning a numerical value between 0 and 1 to each component in the component graph.
2) Compute the significance value for a component ci by:

$$V(c_i) = \frac{1-d}{n} f(c_i) p(c_i) + d \sum_{k \in N(c_i)} V(c_k) w_{ki},$$

With the above approach, the significance values of the components can be calculated by considering the application structure information, the invocation relationships, and the knowledge of component reliability properties in combination [27]. A component with a larger significant value is considered to be more significant. The failures of these significant components will have great impact on other components and thus tend to cause application failures.

c. Fault Tolerance Strategy Selection

Software fault tolerance is widely adopted for critical systems. At the same time, a cloud platform also provides approaches such as virtual machine restart, virtual machine migration, etc. to improve components reliability [18]. By employing these techniques to provide functionally equivalent components, the component failures can be tolerated and thus the overall system reliability can be increased.

IV. Experimental results

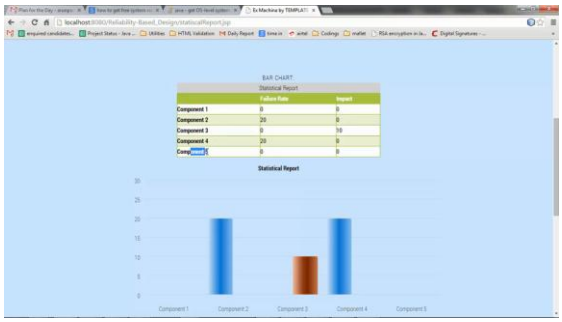
Cloud computing is becoming a mainstream aspect of information technology. A number of tasks have been carried out on cloud computing, including virtualization resource provision and monitoring privacy and trust service level agreement, storage management, data consistency and replication, etc. In recent years, research investigations have been conducted on migrating legacy applications to cloud environment presented a case study of migrating enterprise IT system to IaaS cloud, which illustrated the benefits and major concerns of cloud migration surveyed various approaches for moving legacy system to SOA environment, including wrapping, replacement, etc. We focused on the procedure of migration, strategies on legacy system modernization and methods to improve the cloud platform's reliability, and focuses on the re-design phase during the migration and proposes an optimization framework to improve the cloud application's reliability.

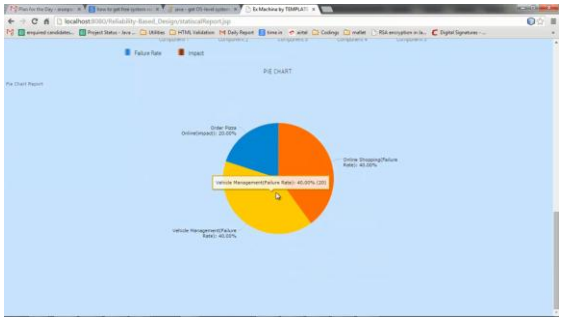
The limitations are Automatically ranking components for legacy applications becomes important, which can aid the designer to optimize the application .The high scalability feature of the cloud makes redundant components easier be obtained. Thus, software fault tolerance becomes a feasible approach to improve the application reliability. At the same time, approaches provided by the cloud platform can also help to build reliable cloud applications.

The main idea of this framework is first to identify significant components whose failures can have great impact on application reliability based on the application structure information and components reliability properties, and then provide fault-tolerant mechanism for these components to improve application reliability.RO Cloud includes two ranking algorithms.

The first algorithm ranks components for the applications that all their components can be migrated to the cloud. The second algorithm ranks components for hybrid applications that only part of their components can be migrated to the cloud. We conduct extensive experiments to evaluate the impact of significant components and their reliability properties on the reliability of the migrated application using reliability information of real world Web services.

The advantages are automatically ranking components for legacy applications becomes important, which can aid the designer to optimize the application. Optimization of the applications based on RO framework makes it more Reliable. Since the critical components are ranked first, the fault tolerance is provided completely, as these critical components all optimized before migration.





V. Conclusion

The reliability-based design optimization framework for migrating legacy applications to the cloud environment framework consists of three parts: legacy application analysis, significant component ranking and automatic optimal fault-

tolerant strategy selection. Two algorithms are proposed in the ranking phase: the first ranks components for the applications where all the components can be migrated to the cloud; the second ranks components for the applications where only part of the components can be migrated to the cloud. In both algorithms, the significance value of each component is calculated based on the application structure, component invocation relationships, component failure rates, and failure impacts. A higher significance value means the component imposes higher impact on the application reliability than others. After finding the most significant components, an optimal fault-tolerant strategy can be selected automatically with respect to the time and cost constraints. The experimental results show that ROCloud1 and ROCloud2 outperform other approaches and can greatly improve the application reliability. In RO Cloud, each component is considered as independent and the fault-tolerant strategy selection is carried out on component basis. In the future, we will study the fault tolerance of interrelated components. In addition, RO Cloud uses the ratios of component failure to application failure to measure the failure impact of components. While the relationship between component failures and application failures can be complicated, more sophisticated models will be investigated in the future work.

VI. future work

Considering more factors when computing the weights of invocations links. Taking the constraint factors such as cost into consideration during the ranking phase, and letting the designer know intuitively which components can make the biggest improvement while cost the least. More experimental analysis on the impact of incorrect prior knowledge such as invocation frequencies and component failure rates.

(1) REFERENCES

[1] D. Denning, Cryptography and Data Security. Addison-Wesley Longman,1982.

[2] Biba, "Integrity considerations for secure computer systems," MITRE Co., technical report ESD-TR 76-372, 1977.

[3] R. Wu, G.-J. Ahn, H. Hu, and M. Singhal, "Information flow control in cloud computing," in CollaborateCom, 2010.

[4] H. Hacig¨um¨us¸, B. Iyer, et al., "Executing SQL over encrypted data in the database-service-provider model," in Proc. 2002 ACM SIGMOD,pp. 216–227.

[5] J. Bacon, D. Evans, et al., "Big ideas paper: enforcing end-to-end application security in the cloud," in 2010 ACM/IFIP Middleware.

[6] P. Mell and T. Grance, "The NIST definition of cloud computing,2011.Available:http://csrc.nist.gov/publications/drafts /800-145/ Draft-SP-800-145 cloud-definition.pdf

[7] I. Foster and C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 2003.

[8] P. Barham, B. Dragovic, et al., "Xen and the art of virtualization," in 2003 ACM SOSP.

[9] T. Ristenpart, E. Tromer, et al., "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds,"pp. 199–212, in Proc.2009 ACM CCS.

[10] Y. Zhang, A. Juels, et al., "Cross-VM side channels and their use toextract private keys," pp. 305–316 in Proc. 2012 ACM CCS,.

[10] Y. Zhang, A. Juels, et al., "Cross-VM side channels and their use to extract private keys,"pp. 305–316 in Proc. 2012 ACM CCS.

[11] A. Ganjali and D. Lie, "Auditing cloud management using information flow tracking," pp. 79–84, in Proc. 2012 ACM STC.

[12] D. Leinenbach and T. Santen, "Verifying the Microsoft Hyper-V hypervisor with VCC," pp. 806–809, in FM 2009: Formal Methods. Springer LNCS 5850, 2009.

[13] M. Dalton, C. Kozyrakis, and N. Zeldovich, "Nemesis: preventing authentication & access control vulnerabilities in web applications," in 2009 USENIX Security Symposium.

[14] D. E. Denning, "A lattice model of secure information flow," CACM, vol. 19, no. 5, pp. 236–243, 1976.

[15] A. Myers and B. Liskov, "Protecting privacy using the decentralized label model," ACM TOSEM, vol. 9, no. 4, pp. 410–442, 2000.

[16] J. Saltzer and M. Schroeder, "The protection of information in computer systems," Proc. IEEE, vol. 63, no. 9, pp. 1278–1308, 1975.

[17] C. S. Alliance, "Security guidance for critical areas of focus in cloud computing," 2011.

[18] J. A. Hall and S. L. Liedtka, "The Sarbanes-Oxley Act: implications for large-scale IT outsourcing," CACM, vol. 50, no. 3, pp. 95–100,2007.

[19] R. T. Mercuri, "The HIPAA-potamus in health care data security,"CACM, vol. 47, no. 7, pp. 25–28, 2004.

[20] V. J. Winkler, Securing the Cloud: Cloud Computer Security Techniques and Tactics. Elsevier, 2011.

[21] CNIL, "Summary of responses to the public consultation on Cloud computing," 2012.

[22] European Commission: Proposal for a General Data Protection Regulation.2012/0011(COD), C7-0025/12, Brussels COM(2012) 11 final,2012.

[23] A. G. Araiza, "Electronic discovery in the cloud," Duke Law & Tech.Rev., 2011.

[24] S. Biggs and S. Vidalis, "Cloud computing: the impact on digital forensic investigations," pp. 1–6,in Proc. 2009 ICITST.

[25] M. Armbrust, A. Fox, et al., "Above the clouds: a Berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[26] J. A. Goguen and J. Meseguer, "Security policies and security models," pp. 11–20, in Proc. 1982 IEEE SOSP.

[27] S. Bleikertz, A. Kurmus, et al., "Secure cloud maintenance: protecting workloads against insider attacks,"pp.83–84, in Proc. 2012 ACM ASIACCS.

[28] D. Suciu, "SQL on an encrypted database: technical perspective," CACM, vol. 55, no. 9, pp. 102–102, 2012.

[29] D. Liu and S. Wang, "Query encrypted databases practically," pp. 1049–1051, in Proc. 2012 ACM CCS,.

[30] C. Cadar, D. Dunbar, and E. Dawson, "KLEE: unassisted and automatic generation of high-coverage tests for complex systems programs," in 2008 USENIX OSDI.