ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2015

A Survey on Optimal Fault Tolerant Strategy for Reliability Improvement in Cloud Migration

Archana Salaskar

II Year ME Student, Dhole Patil College of Engg., Wagholi, Pune, Maharashtra, India.

ABSTRACT: Cloud Computing gives convenient, on-demand network access to shared pool of computing resources like network storage and servers. With On demand use, high scalability and low maintenance cost of Cloud computing more and more enterprises wishes to migrate their legacy application to the cloud environment. Cloud platform itself promises high reliability. And also ensures high quality of service. But due to complicated structure of enterprises application and also large no of distributed components of legacy application the things are becoming complicated. So while migrating any legacy application to the cloud becomes critical and challenging task. To improve reliability proposed system gives ROcloud framework i.e. Reliability- Based optimized framework. ROCloud improves the application Reliability by using Fault tolerance. ROCloudes includes two ranking algorithms. First Ranking Algorithm is used to rank components for the application when all their components of the application will be migrated to the cloud. And second ranking algorithm is used to rank components those are hybrid application i.e. when only part of their components are migrated to the cloud. For components both the Ranking algorithms make use of the Application structure information and historical reliability information. On the basis of ranking results for the most significant components with respect to their predefined components Optimal fault tolerant strategy will be selected automatically.

KEYWORDS: Software Reliability, Cloud Migration, Ranking Algorithm, Fault Tolerance.

I. INTRODUCTION

Cloud computing is a convenient, on-demand network access. It is a shared pool of configurable computing resources. The computing resources like networks, servers, storage, etc., can be provisioned to cloud users on-demand, in the cloud computing environment. It is like the electricity grid. Without the concern of upfront capital or operator expense n companies can deploy their newly developed Internet services to the cloud. Thus, when migrating legacy applications to the cloud environment reliability based optimization is an urgently required research problem [1]. However, cloud computing is not only for newly developed enterprise. Features like cost effective, high scalability and high reliability also attracted other enterprises to migrate their legacy applications to the cloud [2]. First of all enterprises usually have the concern in the cloud environment to keep or improve the application reliability. Traditionally in software reliability engineering, to improve system reliability there are four major approaches: Fault prevention, Fault removal, Fault tolerance, and Fault forecasting.

In the cloud environment, the applications deployed in the cloud environment are usually complicated. And also consist of a large number of components. So only use of fault prevention techniques and fault removal techniques are not sufficient. Software fault tolerance is another approach for building reliable systems. To tolerate faults which can be employ functionally equivalent components. in the cloud environment software fault tolerance approach takes advantage of the redundant resources. And instead of removing faults, it makes the system more robust by masking faults. The cloud platform is flexible and can provide resources on-demand. Still there is a charge for using the cloud components. e.g., the virtual machines of Amazon Elastic Compute Cloud or Simple Storage Service.

Legacy applications usually consist of a large number of components. So providing redundancies for each component is a very expensive task. During the migration of legacy applications to cloud, to assure highly reliability with reduced cost i.e with limited budget, an efficient reliability-based optimization framework is necessary.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 4, April 2015

II. RELATED WORK

In [2] authors illustrates the potential benefits and risks associated with the migration of an IT system in the oil & gas industry from an in-house data centre to Amazon EC2 from a broad variety of stakeholder perspectives across the enterprise, thus transcending the typical, yet narrow, financial and technical analysis offered by providers. The study shows Cloud computing can be a significantly cheaper alternative to purchasing and maintaining system infrastructure in-house. But some drawbacks like cost analysis only focused on system infrastructure costs, and did not quantify the cost of doing the actual migration work; how the support staff costs would be affected by the migration.

Authors gives a novel architectural solution for future cloud service providers based on the concept of Infrastructure as a Service (IaaS) framework and IP network virtualization in[3] . A number of associated schemes have also been designed as building blocks for the proposed framework, including resource description and abstraction mechanisms, virtual network request method and a resource broker mechanism named Marketplace. The proposed framework is able to respond quickly to the infrastructure needs for those cloud services with dynamic resizing of the infrastructure by aggregation or partition to meet capacity requirements of services. At the same time, it improves the utilisation of providers' resources with the creation of an infrastructure incorporating the heterogeneous resources in the data centre. In addition, the proposed marketplace, which also allows the trading of IP network resources between infrastructure providers and cloud service providers, is an important and complementary innovation within the cloud landscape. The proposed framework is able to respond quickly to the infrastructure needs for those cloud services with dynamic resizing of the infrastructure by aggregation or partition to meet capacity requirements of services. At the same time, it improves the utilisation of providers' resources with the creation of an infrastructure incorporating the heterogeneous resources in the data centre. In addition, the proposed marketplace, which also allows the trading of IP network resources between infrastructure providers and cloud service providers, is an important and complementary innovation within the cloud landscape. But there is need of improvement of the provisioning performance of IP infrastructure virtualization for cloud computing.

Authors introduces a probabilistic model and a reliability analysis technique applicable to high-level designs in [4]. The technique is named Scenario-Based Reliability Analysis. Scenario-Based Reliability Analysis is specific for component-based software whose analysis is strictly based on execution scenarios. Using scenarios, we construct a probabilistic model named "Component- Dependency Graph". The Component- Dependency Graphs are directed graphs that represent components, component reliabilities, link and interface reliabilities, transitions, and transition probabilities. In Component- Dependency Graphs, component interfaces and link reliabilities are treated as first class elements of the model. Based on Component- Dependency Graphs, an algorithm is presented to analyze the reliability of the application as the function of reliabilities of its components and interfaces. A case study illustrates the applicability of the algorithm. The Scenario-Based Reliability Analysis is used to identify critical components and critical component interfaces, and to investigate the sensitivity of the application reliability to changes in the reliabilities of components and their interfaces. Cloud computing is becoming a mainstream aspect of information technology. More and more enterprises deploy their software systems in the cloud environment.

The cloud applications are usually large scale and include a lot of distributed cloud components. Building highly reliable cloud applications is a challenging and critical research problem. To attack this challenge, author propose a component ranking framework, named FTCloud, or building fault-tolerant cloud applications in[6]. FTCloud includes two ranking algorithms. The first algorithm employs component invocation structures and invocation frequencies for making significant component ranking. The second ranking algorithm systematically fuses the system structure information as well as the application designers' wisdom to identify the significant components in a cloud application. After the component ranking phase, an algorithm is proposed to automatically determine an optimal fault-tolerance strategy for the significant cloud components. The experimental results show that by tolerating faults of a small part of the most significant components, the reliability of cloud applications can be greatly improved.

III. PROPOSED ALGORITHM

A. *Automated significance ranking algorithms:*

i) Component Ranking for ordinary application:

- Initialize by randomly assigning a numerical value between 0 and 1 to each component in the component graph.
- Compute the significance value for each component.
- The significance values can be calculated either iteratively or algebraically. The iterative method is repeating the computation until all significance values become stable.

ii) Component Ranking for Hybrid application:

- The components of a hybrid application are divided into two sets by their nature. One set for the components deployed in a private data center, denoted as P, and the other for the components moved to the cloud, denoted as C.
- For each component calculate the significance value.
- The significance values can be calculated either iteratively or algebraically. The iterative method is repeating the computation until all significance values become stable.

B. *Fault tolerance strategy selection algorithm:*

- First, the aggregated failure rate f, response-time t, and the resource cost r of each fault tolerance strategy candidate are calculated by using RB, NVP, Parallel and VM restart. And the strategies which could not satisfy the response-time constraints will be removed.
- Second, list the Top-K significant components according to the descending order of their significance value.
- Third, the strategy with minimum resource cost will be selected for each of the components as their initialization strategy to make sure all of them are fault-tolerant. Then for each component, select the candidate with the lowest aggregated failure rate as the optimal one. By repeating the last step until it meets the user resource cost constraints, the reliability-based design optimization can be achieved.

IV. SYSTEM DESIGN PROCSS

Reliability optimization framework i.e. ROCloud, includes three phases:

- Legacy application analysis;
- Automated significance ranking; and
- Fault tolerance strategy selection.

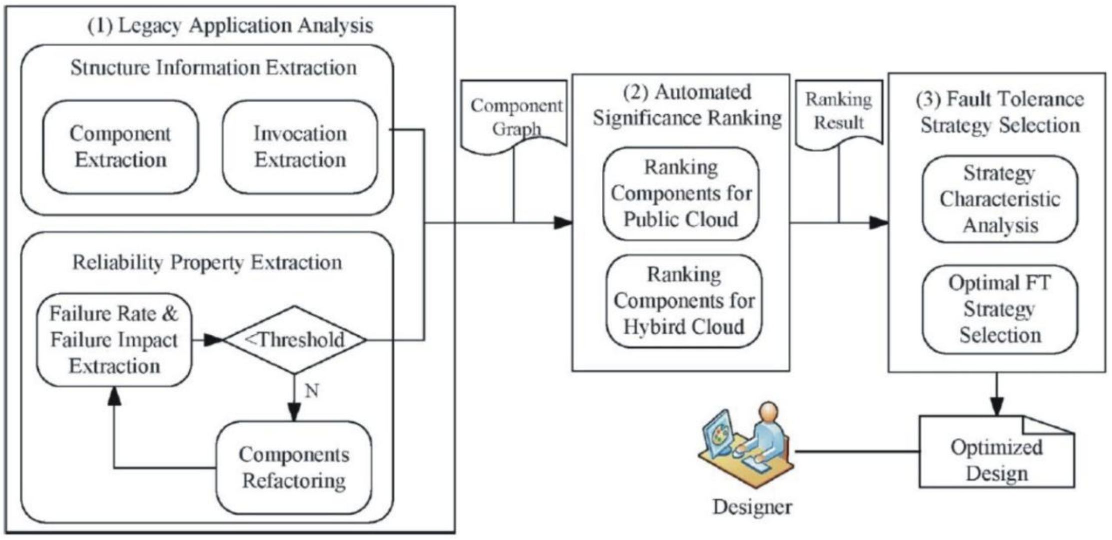
(1) Legacy Application Analysis
Structure Information Extraction
Component Extraction
Invocation Extraction
Reliability Property Extraction
Failure Rate & Failure Impact Extraction
<Threshold
N
Components Refactoring
Component Graph
(2) Automated Significance Ranking
Ranking Components for Public Cloud
Ranking Components for Hybird Cloud
Ranking Result
(3) Fault Tolerance Strategy Selection
Strategy Characteristic Analysis
Optimal FT Strategy Selection
Optimized Design
Designer

Fig.1. System Architecture [1]

Legacy application analysis:

Both structure and failure information are extracted during the legacy application analysis phase. The structure information extraction consists of two subprocesses:

Component extraction:

The structure information includes components and the invocation information. The components are extracted from legacy applications by source code and documentation analysis. 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.

Invocation extraction:

Component failure rate and failure impact collection: The failure rate and failure impact information can be collected from the execution logs or test results of legacy applications. 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.

Automated significance ranking:

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.

Software fault tolerance strategies:

Recovery Block (RB), N-Version Programming (NVP) and Parallel are three widely used strategies in software fault tolerance. Since RB strategy invokes standby components sequentially when the primary component fails, its response time is the summation of the execution time of all failed versions and the first successful one. NVP strategy needs to wait for all n responses from the parallel invocations to determine the final result, thus its response time depends on the slowest version. While Parallel strategy employs the first returned response as the final result, its response time is the minimum one of all replications. So it can be concluded that the response time performance of RB is generally worse than that of NVP, which in turn is worse than that of the parallel strategy. Since NVP and Parallel use parallel component invocations and all the resources need to be allocated before the execution, while in RB extra resources will be allocated only when the primary component fails, the required resources of NVP and Parallel are much higher than those of RB. All three strategies can tolerate crash faults, and NVP strategy can also mask value faults.

Strategies based on cloud features:
Cloud platforms often provide approaches such as virtual machine restart, virtual machine migration etc. to improve reliability. These approaches can also tolerate crash faults. The strategy based on virtual machine (VM) restart is similar to the RB strategy.

Strategies based on cloud features can also improve components reliability by tolerating crash faults. They have much lower demand on extra resource compared to the software fault tolerance strategies, while they have considerable overheads which can increase the response time. Different strategies have different resource requirement and different effects on response time. RB strategy can affect the response time and resource allocation if there is a failure. While the parallel and NVP strategies have little effect on the response time but will affect the resource allocation in all cases. The virtual machine restart strategy will not affect resource allocation but can affect the response time if there is a failure. Employing a suitable fault tolerance strategy for the significant components can help achieve optimal resource allocation while improving application reliability. Each fault tolerance strategy has a number of variations, thus selecting an optimal strategy for each significant component is time consuming. An automatic optimal fault tolerance strategy selection algorithm is therefore required to reduce the workload of application designers.

Four candidates are employed for fault tolerance which include recovery block, N-version programming, parallel, and virtual machine restart. These strategies can be employed to tolerate crash and value faults.
Other types of fault tolerance mechanisms can be added to ROCloud without fundamental changes.

V. CONCLUSION AND FUTURE WORK

Authors presents a reliability-based design optimization framework for migrating legacy applications to the cloud environment. They proposes a component ranking framework for fault-tolerant cloud applications. In proposed component ranking algorithms, the significance value of a component is determined by the number of components that invoke this component, the significance values of these components, how often the current component is invoked by other components, and the component characteristics. After finding out the significant components, System proposes an optimal fault-tolerance strategy selection algorithm to provide optimal fault-tolerance strategies to the significant components automatically, based on the user constraints.

In the proposed system, only study of the most representative type of software component graph is done, i.e., scale-free graph. Since different applications may have different system structures, we will investigate more types of graph models. Future work also includes: Consideration of more factors (such as invocation latency, throughput, etc.) when computing the weights of invocations links and investigating the component reliability itself besides the invocation structures and invocation frequencies and more experimental analysis on real-world cloud applications and also more investigations on the component failure correlations.

REFERENCES

[1] Weiwei Qiu, Zibin Zheng, Member, IEEE, Xinyu Wang, Xiaohu Yang, "Reliability Based Design Optimization for Cloud Migration" in 1939-1374 _ 2013 IEEE.
[2] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville, ''Cloud Migration: A Case Study of Migrating an Enterprise IT Sstem to IaaS,'' in Proc. IEEE 3rd Int. Conf. CLOUD, 2011, pp. 450-457.
[3] Bo Peng, Ali Hammad, Reza Nejabati, Siamak Azodolmolky "Network Virtualization Framework for IP Infrastructure Provisioning " in 978-0-7695-4622-3/11.
[4] Sherif M. Yacoub, Bojan Cukic, and Hany H. Ammar " Scenario-Based Reliability Analysis of Component-Based Software" in DAAH04-96-1-0419, monitored by the Army Research Office, and by the NASA research grant No. NAG4- 163, administrated by the Institute for Software Research.
[5] Zibin Zheng and Michael R. Lyu " A Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services" 978-0-7695-3310-0/08 .
[6] Zibin Zheng, Tom Chao Zhou, Michael R. Lyu, and Irwin King, "Component Ranking for Fault-Tolerant Cloud Applications " in 1939-1374/12.
[7] Z. Zheng, Y. Zhang, and M.R. Lyu, ''CloudRank: A QoS-Driven Component Ranking Framework for Cloud Computing,'' in Proc. Int'l SRDS, 2010, pp. 184-193.

[8] Z. Zheng, T.C. Zhou, M.R. Lyu, and I. King, ''FT Cloud: A Ranking-Based Framework for Fault Tolerant Cloud Applications,'' in Proc. ISSRE, 2010, pp. 398-407.
[9] Liang Zhou CloudFTP: A Case Study of Migrating Traditional Applications to the Cloud" in IEEE 978-0-7695-4923-1/12

BIOGRAPHY

Archana Kamlakant Salaskar is a Research Scholar in the Computer Engineering Department, Dhole Patil College of Engg., Pune, MS, India. She is perusing Master of Computer Engineering (ME) degree from the same organisation. Her research interests are Network Security, Data Mining, Cloud Computing, Web etc.