THROUGHPUT VARIANT COMPONENT RANKING IN DYNAMIC FTCLOUD FRAMEWORK

BY

T. SUDHESHNA * C. SHOBA BINDU **

* Department of CSE, JNTUA College of Engineering, Ananthapuram, India
** Associate Professor & Head, Department of CSE, JNTUA College of Engineering, Ananthapuram, India

ABSTRACT

FTCloud is an emerging cloud paradigm that orchestrates multiple cloud technologies and is becoming the main stream aspect of providing service. As software, Fault Tolerance (FT) mechanisms mask failures earlier to improve reliability. To address this challenge, Zibin Zheng proposed a component ranking framework with fault tolerance named FTCloud to tolerate failures in software. In FTCloud more characteristic factors like throughput and dynamic fault-tolerance mechanisms are not implemented. To ensure reliability ' Dynamic FTCloud ' framework mainly concentrates on throughput with random graph model in FTCloud1 while employing response time for services. The FTCloud2 focuses on failure probability of components as extension to the FTCloud. In this paper, dynamic optimal fault-tolerance strategy is implemented in the framework along with the previous algorithm of design diversity techniques .The prospecting results show that tolerating faults of significant components are having enormous improvement with reliability.

Keywords: Dynamic FT Strategy, Throughput, Cloud Application, Fault Tolerance.

INTRODUCTION

The cloud computing phenomenon is the backbone of various internet services (computing, storage, data access etc.) in which end users do not depend on physical locations of their system. The key concept of cloud computing is to give low cost unit of computing pool from the shared resources which are distributed in different places[1].Nowadays cloud provides infrastructure management for each service . The advantage of cloud computing is to provide services on demand basis with pay and use concept. For providing the composition of services, it is required to combine all legacy services into a component where availability becomes a major concern [15], but sometimes the services become unavailable due to fault occurrence. Many giant companies like Amazon, Google, and Microsoft, web hosting companies such as Rack space and GoGrid, and new start-ups such as Flexiant and Heroku are becoming service providers. A practical challenge thus arises which is to provide services without wastage of time and data.

To analyze the availability of cloud services, assessment of components is a reasonable solution. As cloud can host applications as Software as a Service (SaaS), many applications are being deployed in cloud [2].However, ease of availability, maintenance and reliability [4] is becoming complex, while providing services. Hence multiple redundant components are going to be copied at various distributed places .One of the reasons for the unavailability of cloud components is the lack of fault tolerance [8].To provide services through components, building highly reliable environment is a challenging task. In order to build reliable software, the corresponding engineering disciple in traditional approaches uses fault tolerance, fault removal, fault prevention and fault forecasting [6]. Software fault tolerance techniques provide protection against errors in translating the requirements into components, but they do not provide explicit protection against errors. Design diversity [7] is a famous fault-tolerance technique, where the components are designed to tolerate faults. It is an identical service through separate implementation where diversity in the design of software is independent.

However, developing fault tolerant components is a costly affair and thus it is applied only for machine critical applications such as spare research systems, flight control systems etc. As cloud applications have large number of components, it is really expensive to develop fault tolerant components. In order to reduce the effort and time to make the system robust, only the components which are critical are to be identified and for them only fault tolerance has to be built. As reported by Microsoft, it is important to fix top 20 percent bugs with respect to critical components which can avoid 80 percent of crashes and failures [5]. Based on this 80-20 rule, Zheng et al [25] proposed a component ranking framework by name "FTCloud" which makes the cloud applications reliable [6] as they are fault tolerant. It identifies critical components and makes them fault-tolerant. Its two ranking algorithms identify critical components and apply appropriate fault tolerant strategy to ensure the best performance of cloud applications. However, FTCloud can be further enhanced by considering throughput characteristics of the components in the cloud applications. To focus on reliability, our contributions include:

- Making use of throughput to calculate the amount of work that applications perform concurrently at runtime. It also enables the framework to choose the best component.
- Proposed adaptive n-version algorithm includes the weight factor at voting part where it increases reliability at faulty components.
- Fuzzy voting algorithm determines the correct output component based on different voting methods to decrease the latency on innovation.

The rest of the paper is organized as follows. Section 1 describes related fault tolerant cloud applications literature. Section 2 introduces the proposed component ranking framework. Section 3 specifies the Throughput significant ranking. Section 4 implements Dynamic FT Strategy. Section 5 analyzes the experimental results.

1. Related Work

The traditional software reliability engineering concentrates in demand customer's perspective and fault tolerance is widely used for building multiple redundant copies and reliable applications [9]. There are many fault tolerance techniques to obtain reliability by preventing fault occurrence in their phases. FT-Software provides service complying with the relevant specification inspite of faults. The FT techniques include distributed recovery blocking [10], N-version programming [11], N self-checking programming [12]. The fault tolerance strategies are classified into active and passive strategies based on the replication of redundancy components [13].The applications such as WS-Replication [22], FTWeb[23] send requests at the same time to various replicas and accept first response as final result. Passive strategy sends request in sequence, while primary web services use final result as response such as FT-SOAP[24].Software fault tolerance is considered as a feasible approach for building high reliable cloud computing environment. However, making such components reliable and fault tolerant is very important. Zheng et al. [25] proposed a component ranking framework "FTCloud" which is used to build fault tolerant cloud applications.

Nowadays, significant research problem is the design issues to invoke and rank components [17] with weight calculation for high reliability QoS. However, our approach in this paper is influenced by design [25] which is used to improve the component ranking framework by considering more fault tolerant approaches to select a critical component in a significant manner. However, Dynamic FTCloud framework contributes to enhance robustness and reliability for building cloud applications

1.1 Demerits of FTCloud

1. Identification of significant component through normal random graph is difficult.
2. Different Dynamic FT-Strategies are not included to specify critical component.
3. In cloud environment, throughput is not considered to define invocation structures.

To address the above issues, we propose a component ranking including throughput which ranks the significant component dynamically. Next, for acquired components, dynamic FT Strategies are applied by which optimal fault tolerance strategy is dynamically suggested for application designers.

2. Proposed Component Ranking Framework

Component rank model is a repository of software component libraries which are off-the-shelf programs with a novel graph-representation at the end of result. Often, used components are ranked as prior so that designers have quick access to that component. Components are divided into critical and noncritical components in clouds which reduce the fault tolerance for providing reliable services to IaaS users that is mainly focused in this paper.

In the proposed framework, component ranking structures and use of dynamic fault strategies Dynamic FTCloud enhances the performance of component ranking framework by.

- Including characteristics like throughput in component ranking framework.
- Identification of significant component failures for fault tolerance which is easy with random graphs.
- Challenging task for the application designer which is selection of fault-tolerance strategies and to find out optimal results, dynamic FT-Strategy is implemented.

By the implementation of the above framework, the designers of cloud applications can build dynamic, highly reliable and robust system which is extremely fault tolerant.

2.1 System Architecture

Dynamic FT- framework includes throughput based ranking and dynamic optimal FT selection [8].Dynamic FTCloud components are ranked by invocation relationship and performance is evaluated by designers to design the structure of cloud components.

Figure 1 shows the significant components to be identified, arrow specifies input, dotted line specifies output of the application and intermediate results are recorded at every step of transition in document.

3. Throughput Significant Ranking

Cloud computing is the internet based service provider

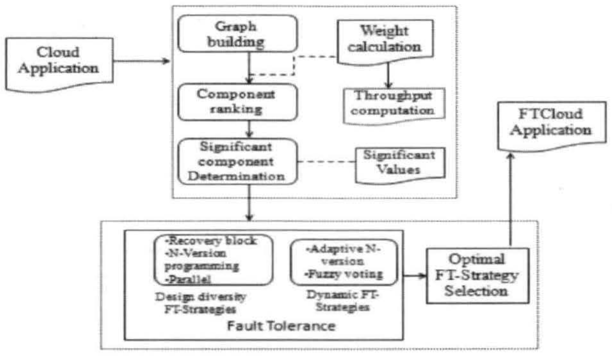

Figure 1. Architecture of the Dynamic FTCloud framework

where each software component interacts with other components as nodes. These nodes are represented in the form of graph with internet connections as edges. By taking component graph as input which is proposed by Michael R.Lyu [25], performance of the service is to be measured by throughput, which is the solution to FT Cloud1.

The throughput, TH is the average of the output services provided by a component per unit time, e.g., number of services through internet per hour. Being time-dependent, throughput is calculated by a component as a set of services/messages and by using the following equation,

TH = number of service requests completed / time taken to complete the service

The main goal is to get the performance of cloud components services with different concurrent cloud components. They are many tools existing for the performance evaluation. Examples are JMeter, LoadUI, java bench, etc.

The above services is compared with the performance of the services output as shown in Figure 2. As user services

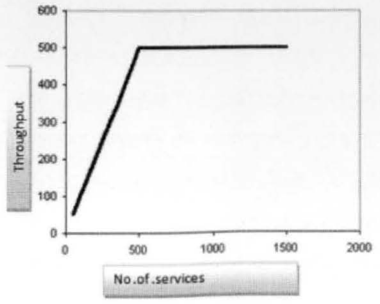

Figure 2. Throughput analysis

Compon-ent FP	Methods	Recovery block	N-Version	Parallel	Adaptive N-version	Fuzzy voting
0.4%	FTCloud1	0.117	0.4115	0.020	0.0081	0.020
	FTCloud2	0.117	0.117	0.020	0.00408	0.0102
0.5%	Proposed	0.037	0.037	0.14285	0.02040	0.006802
	FTCloud1	0.055	0.055	0.00936	0.00374	0.00936
	FTCloud2	0.05	0.117	0.009365	0.0018730	0.004682
	Proposed	0.083	0.0833	0.096	0.009365	0.003121
0.6%	FTCloud1	.25	0.25	0.00071	0.003086	6.20181*10^-9
	FTCloud2	0.25	0.25	0.00714	2.143*10^-6	1.24*10^-9
	Proposed	.166	0.1666	0.00001605	4.46*10^-7	2.236*10^-7

Table 1. Impact of Application Failure Probabilities

increase, throughput gradually increases upto a certain threshold point after which services are provided in same throughput order.

4. Dynamic Fault Tolerance Strategies

Sustainability of fault tolerance is crucial for critical components where there is no significance for non-critical components.

There are many fault tolerant strategies proposed in [19] which are design dependency of structure. Example: Recovery Block (RB), N-Version Programming (NVP) and Parallel Programming. By these techniques, the dynamic reliability in cloud cannot be achieved. Hence we implement dynamic FT-strategies for FTCloud1 and FTCloud2 for decreasing failure probability as shown in Table 1. Ex: Adaptive N-version Programming, Fuzzy voting.

4.1 Adaptive N-version Programming

It is similar to N- version where an individual weight factor for all component version and actual, uptime utilization of adaptive services[14] are included [20]. Then, based on the maximum capacity of weight factor, the voting procedure is conducted as shown in Figure 3. In the

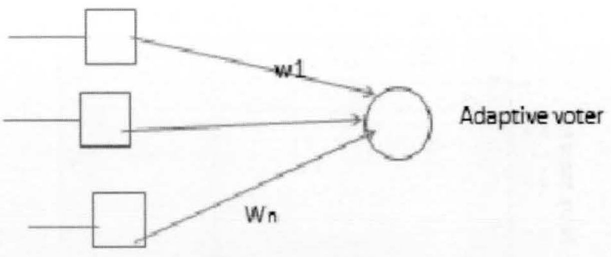

Figure 3. Adaptive N- Version

component-based strategies, for building the individual versions of the system component, throughput is considered. Here N-version programming is defined as the functionally generated independent equivalent programs of N>=2 from same initial specification.

$$R_{\mathrm{mod},I,n} = \prod_{i=1}^{I} R_{mi,n} \qquad (1)$$

Where $R_{mi,n}$ is the reliability of module stage I comprising n version modules. Failure probability is identified based on equation 1 as it is subtracted from one giving the failure rate.

4.2 Fuzzy Voting

In this, correct output is selected from different outputs which are obtained from various redundant software versions. Traditional voting method is based on an output classification of disjoint subsets [16]. This is similar to the N version programming with

- Majority voting (NVP-MV)
- Consensus voting (NVP-CV)
- Maximum likelihood voting (MLV)

Fuzzy relation is the degree of interconnecting set of elements that comprises the relation [18]. The fuzzy equivalence relation exists if and only if all properties of reflexivity, symmetry, transitivity are satisfied.

This paper focuses on dynamic fault tolerant strategies. The performance comparison of four fault tolerance strategies are presented in Table 2.

	RB	NVP	Parallel	Adaptive N - Version	Fuzzy Voting
Response Time	Middle	Middle	Good	Good	High
Required Resources	Middle	High	High	Middle	Low
Fault Tolerance	Crash	Crash, Value,	Crash	Crash, Value, reliability	Crash

Table 2. Comparision of FT Strategies

5. Experimental Results

The experimental solution for this application is built in Java programming language. The application was developed by java frames where the database used is Xampp server. Pajek tool [21] is used to model various

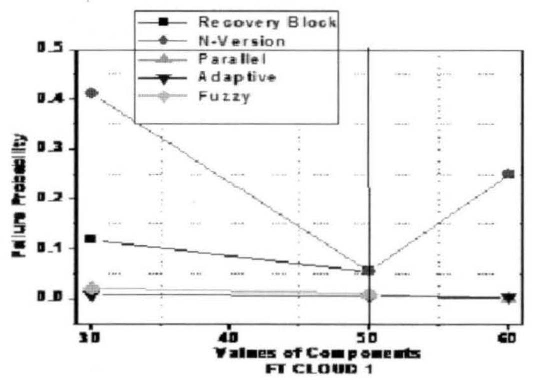

Figure 4. Impact of Component Failure Probability in FTCloud1

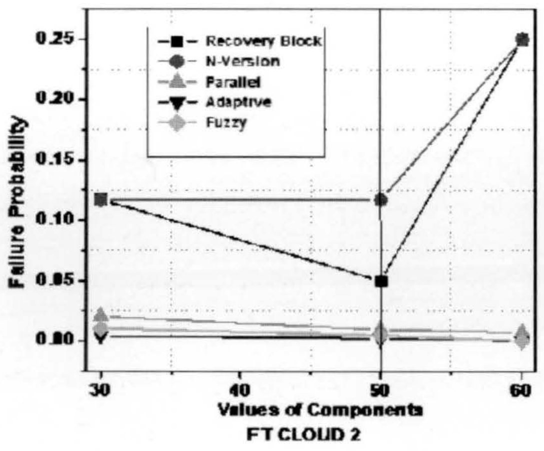

Figure 5. Impact of Component Failure Probability in FTCloud2

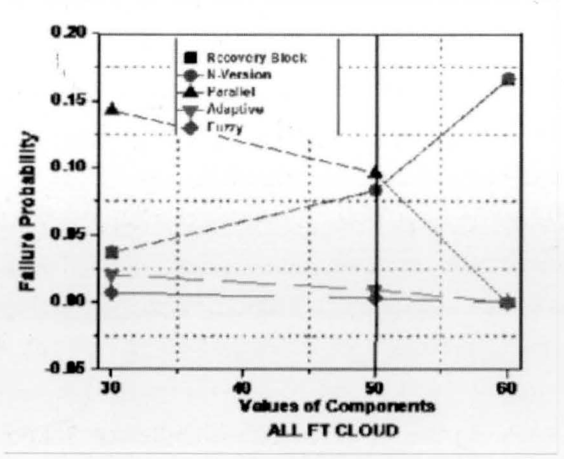

Figure 6. Impact of Component Failure Probability in AllFTCloud

cloud application components. After generation, the values are going to take out for specification of throughput and failure probability. The impact of failure probability of cloud applications with respect to fault tolerant strategies and component rankings are represented in Table2.

The dynamic component ranking algorithms namely FTCloud1 and AllFTCloud include throughput where FTCloud2 includes extra FT Strategies .The results are visualized as shown in the graphs.

In Figure 4, the resultant graph performance of all strategies are shown expect N-version, remaining are linear because it include ranking of significant component structure in serial while parallel structuring is followed in N-version.

Figure 5 shows that, the failure probability is tolerated by including FT Strategies only for critical components. In recovery block and N-version, as components increase, the failure probability increases because it is a static strategy where parallel includes the throughput for responding so that its probability is decreased, while Adaptive N-version and fuzzy include the throughput dynamically where probabilities are linear even when the components are increased in cloud environment.

Figure 6 shows that, all strategies are applied and are included for all components in the cloud system. So, performance drastically changes due to failure probabilities which are seen for all strategies. Throughput is included for every component for getting response time.

To study the impact of dynamic component ranking approach of failure probabilities on system performances, each dynamic FT-strategy with FTCloud1, FTCloud2 and All FT Cloud by impact factor as number of components 50 is taken as 'top-k' which is necessary for analyzing the results. In this, FTCloud2 shows variant resultant curve where the performance is highly increased. When All FT Cloud is applied, then the impact of components on every strategy is identified clearly. The above results show that FTCloud2 and All FT Cloud achieve the best performance.

Conclusion

The "Dynamic FTCloud" framework is used to build highly reliable fault-tolerant distributed cloud applications. For providing services to user's component ranking framework, one can employ not only tolerance towards crashes and faults, but also identify the malicious component on the asynchronous environment.

To gain more insight, proposed through that impacts ranking of components with new techniques of optimal fault tolerance strategies. Hence the failure probability at each redundant component levels is decreased and reliability is increased. In this work the throughput is taken as parameter for providing service and ranking of components. By the random graph model in cloud environment, performances of components are increased. The empirical results revealed that the proposed framework is more robust and very highly fault tolerant.

References

[1]. "Cloud Computing in wikinvest",http;//wikiinvest.com/concept/Cloud Computing.

[2]. "Software Fault Tolerance"-Reliable Software Technologies-Ada-Europe 2003. *Lecture Notes in Computer Science,* Volume 2655, pp 45-67, 2003.

[3]. **Elmendorf,w.r.,(1972).** "Fault-Tolerant Programming," *Proceedings of FTCS-2*, Newton, MA, pp. 79–83.

[4]. **kanoun, K., et Al.,(1993).** "Reliability Growth of Fault-Tolerant Software," *IEEE Transactions on Reliability,* Vol. 42, No. 2, pp. 205–129.

[5]. **P. Rooney (2002).** "Microsoft's CEO: 80-20 Rule Applies to Bugs, Not Just Features," ChannelWeb.

[6]. **S.S.Gokhale and K.S. Trivedi (2002).** "Reliability Prediction and Sensitivity Analysis Based on Software Architecture," *Proc. Int'l Symp. Software Reliability Eng.* (ISSRE '02), pp. 64-78.

[7]. **LYU, M. R. (ed.) (1995).** *Software Fault Tolerance*, New York: John Wiley & Sons.

[8]. **Z. Zheng and M.R. Lyu (2008).** "A Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services," *Proc. Sixth Int'l Conf. Web Services*, pp. 145-152 .

[9]. **S. Gorender, R.J. de Araujo Macedo, and M. Raynal (2007)** "An Adaptive Programming Model for Fault-Tolerant Distributed Computing," *IEEE Trans. Dependable and Secure Computing,* Vol. 4, No. 1, pp. 18- 31, Jan.-Mar. 2007.

[10]. **B. Randell and J. Xu (1995).** "The Evolution of the Recovery Block Concept," *Software Fault Tolerance, M.R. Lyu, ed.*, pp. 1-21, Wiley,1995.

[11]. **A. Avizienis (1995).** "The Methodology of N-Version Programming," *Software Fault Tolerance, M.R. Lyu, ed.*, pp. 23-46, Wiley,1995.

[12]. **J. Laprie, J. Arlat, C. Beounes, and K. Kanoun (1990).** "Definition and Analysis of Hardware- and Software-Fault-Tolerant Architectures," *Computer,* Vol. 23, No. 7, pp. 39-51.

[13]. **Z. Zheng and M.R. Lyu (2008).** "A Distributed Replication Strategy Evaluation and Selection Framework for Fault Tolerant Web Services," *Proc. Sixth Int'l Conf. Web Services*, pp. 145-152.

[14]. **D. Ardagna and B. Pernici (2007).** "Adaptive Service Composition in Flexible Processes," *IEEE Trans. Software Eng.*, Vol. 33, No. 6, pp. 369-384, June 2007.

[15]. **S. Brin and L. Page (1998).** "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Proc. Int'l Conf. World Wide Web*.

[16]. **Z. Tong and R. Kain (1991).** "Vote assignments in weighted voting mechanisms". *IEEE Transactions on Computers,* Vol(40), pp. 664-667,May 1991.

[17]. **Zibin Zheng, Xinmiao Wu, Yilei Zhang, Michael R. Lyu, Jianmin Wang (2013).** "QoS Ranking Prediction for Cloud Services," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 24, No. 6, pp. 1213-1222, June 2013.

[18]. **Y.W. Leung (1995).** "Maximum Likelihood Voting for Fault Tolerant Software with Finite Output Space", *IEEE Trans. Rel,* Vol. 44(3), pp. 419-427.

[19]. **Zaipeng Xie, Hongyu Sun and Kewal Saluja.** "A Survey of Software Fault Tolerance Techniques".

[20]. **Avizienis, A (1997).** "On the Implementation of NVersion Programming for Software Fault- Tolerance During Execution," *COMPSAC '77,* Chicago, IL, pp. 149–155.

[21]. **Vladimir Batagelj and Andrej Mrvar.** *"Pajek –*

Program for Large Network Analysis" http://vlado.fmf.uni-lj.si/pub/networks/pajek/.

[22]. **J. Salas, F. Perez-Sorrosal, M. Patin˜ o-Martı´nez, R. Jandime´nez- Peris (2006).** "WS-Replication: A Framework for Highly Available Web Services," *Proc. 15th Int'l Conf. World Wide Web*, pp. 357-366.

[23]. **G.T. Santos, L.C. Lung, and C. Montez (2005).** "FTWeb: A Fault Tolerant Infrastructure for Web Services,"*Proc. IEEE Ninth Int'l Conf. Enterprise Computing*, pp. 95-105.

[24]. **Q.Z. Sheng, B. Benatallah, Z. Maamar, and A.H. Ngu (2009).** "Configurable Composition and Adaptive Provisioning of Web Services", *IEEE Trans. Services Computing*, Vol. 2, No. 1, pp. 34-49, Jan-Mar, 2009.

[25]. **Zibing Zheng and M.R.Lyu (2011).** "Component Ranking for Fault-Tolerant Cloud Applications", *IEEE Transaction on Service Computing.* Vol.5(4).

ABOUT THE AUTHORS

T. Sudheshna is currently pursuing her M.Tech in Software Engineering at Jawaharlal Nehru Technological University, Anantapur. She received her B.Tech Degree in Information Technology from G.Pullaiah College Of Engineering & Technology, Kurnool, Andhra Pradesh, India in 2011. Her research interests are in the fields of Service Computing, Cloud Computing.

C. Shoba Bindu is currently working as a Professor and Head in the Department of Computer Science & Engineering at Jawaharlal Nehru Technological University, Anantapur. She obtained her Bachelor's Degree in Electronics and Communication Engineering, Masters and Ph.D. degrees in Computer Science and Engineering from Jawaharlal Nehru Technological University, Anantapur. She has published several research papers in National, International Conferences and Journals. Her research interests includes Network Security and Wireless Communication Systems.