THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Improving the Performance of Fault Tolerant Mechanisms by Optimal Strategy Selection in Cloud

Ginu Chacko
B.E. CSE Student, SNS College of Technology, Coimbatore, TN, India
S. Indhujaa
B.E. CSE Student, SNS College of Technology, Coimbatore, TN, India
A. Abuthahir
B.E. CSE Student, SNS College of Technology, Coimbatore, TN, India
S. Vidya
Assistant Professor, SNS College of Technology, Coimbatore, TN, India

Abstract:
In Cloud Computing technology the most scalable services are effortlessly utilized in an on-demand method by way of internet. The software Fault Tolerance technology is adopted far and wide to magnify the overall system trustworthiness in critical applications. System trustworthiness can be enriched by having the constituents that are more or less equivalent to tolerate the constituent breakdown. We commence three renowned Fault Tolerant strategies in our projected logic. The enhanced policies care about the Optimal Fault Tolerant Strategy Selection algorithm to bestow the scalable and bendable Cloud applications.

Key words: *Cloud Computing, Fault Tolerance, Optimal Strategy Selection*

1. Introduction

Cloud Computing foretells a most important transformation in how we warehouse the information and accomplish the applications. As a replacement for implementing the Cloud, applications progression on a desktop computer, the whole thing can be hosted on Cloud that is an assortment of computer systems and servers which can be utilized by means of the internet [7]. Cloud Computing enables us to access every applications and documents from anywhere across the globe, freeing us from the boundaries beyond the desktop and makes it simple for crew members in unique locations to team up [12]. The advent of Cloud Computing is the working out equivalent of the electricity rebellion of some centuries ago. Before the invention of electrical wares, each and every industries and business people fabricated their own source of electricity by means of generators. After the electricity came into existence, many industries put down their generators and procured the means of electricity from other sources, at an economically low cost (reliable and flexible) means than they could make out by their own way.

In Cloud Computing the users who have the authorization can, not only access the documents but also can make changes and combine those documents in real time basis. The Cloud of computer systems expands further than a single company or enterprise. Customers from various locations in and around the organization, and from various other organizations, wished to work together on projects that covered the company and environmental related boundaries. Cloud Computing is an upcoming conventional feature of Information Technology. More progressively enterprises organize their computer software systems in the Cloud environment. The applications on Cloud are usually large scale and contain a lot of disseminated Cloud constituents. Building Cloud applications are highly trustworthy for challenging and dangerous research issues. Information processing systems have enlarged the significance of its correct and uninterrupted operation even in the incidence of faulty constituents. To address and focus on this particular issue, we propose a Cloud skeleton to build Fault Tolerant Cloud applications.

2. Related Work

In [1] the authors have discussed about the Fault Tolerance Delivery Scheme where the knowledge about the Fault Tolerance for the users is not needed. A comprehensive high-level approach that provides the implementation level details of the momentous Fault Tolerance techniques to Cloud application developers and users by way of a dedicated and efficient service layer.

In [2] Low Latency Fault Tolerance mechanism is used where the web services can be replicated by means of a strong replication mechanism without any modifications to the applications. Crash fault and timing faults were considered in this paper.

In [3], the authors discuss about the fault taxonomy and what is the need of Fault Tolerance including its various methodologies for implementing Fault Tolerance.

The paper [4] describes the basic model for Fault Tolerance in Cloud Computing. This Fault Tolerance model tolerates the failures on the basis of trustworthiness of each computing host. A computing host is selected for computation on the basis of its trustworthiness and can be eliminated, if it does not perform well for the applications.
In [5] the authors describe about adaptive Fault Tolerance in real time Cloud Computing. Here Fault Tolerance decision making is done based on the trustworthiness of processing nodes. Trustworthiness assessment that is provided is only for three virtual machines.

3. Fault Tolerance Strategies
The Fault Tolerant strategies have number of discrepancies based on different compositions. For each and every constituent in a Cloud based application, the Fault Tolerance strategy discrepancies that are identified will be taken as the candidates and the optimal one needs to be identified. For each momentous constituent that is in need of a Fault Tolerance strategy, the designer can specify some constrictions (e.g., Response time of the constituent has to be smaller say 1,000 milliseconds, etc.). Two user constructions are considered: one for response time and one of cost. Three renowned Fault Tolerance strategies are pioneered with formulas for computing the failure probabilities for the Fault Tolerant modules. Failure probability is the probability that the particular constituent will fail. The value of this computed failure probability is taken in the scale of [0, 1].

- Recovery Block
- N- version Programming
- Parallel

3.1. Recovery Block (RB)
This method is akin to the cold backup method for the one that is about hardware Fault Tolerance. Normally, in this method multiple discrepancies of software which are equivalent to that of the functionality are deployed in a timely redundant manner. The test of acceptance will be carried out that is used to test the soundness of the outcome produced by the primary version.
Suppose if the result from the primary version that obtained passes the test of acceptance, this outcome will be reported and execution stops. And if on the other hand, the output from the primary version fails the test of acceptance, another version from among the vibrant versions is used and the outcome produced is checked by this means of test.

3.2. N-Version Programming (NVP)
In this methodology, independently generated programs that are being functionally correspondent called versions are executed in parallel. A logic that is of majority voting is used to compare the outcomes produced by each and every versions and report one of the outcomes which is assumed to be correct.
The ability to withstand the faults depends on how independent the unique versions of the constituents are. This criterion has been applied to a large number of real life based systems like railroad traffic control and air flight control, even though the overhead cost involved in finding different versions of programs and succeeding the voting logic may be high [11].

3.3. Parallel
Parallel strategy invokes all the constituents that are functionally correspondent in parallel and the first returned response will be employed as the last result. A parallel module fails only if the entire redundant constituents fail.

4. Problem Definition
Fault Tolerance is the main issue discussed in this paper. Fault Tolerance is the ability of each functional unit in the system to continue to perform consistently a required functionality even in the incidence of failure or crash. Fault on the Cloud leads to performance issue. The traditional way of achieving trustworthy and highly available software is to exploit the efficient Fault Tolerance methods at development time [8]. Applications organized in Cloud Computing infrastructure can acquire the required Fault Tolerance possessions from an intermediary service provider. So we focus to employ a suitable Fault Tolerant strategy for the Cloud constituents such that it is important to achieve the optimal Cloud application design. Since the Cloud applications normally include a more number of constituents, it is still too costly to provide alternative constituents for all the means of Cloud constituents. There is no need to provide Fault-Tolerance strategies for the non momentous type of constituents, where the failures have limited impact on the systems. To reduce the economic crisis we need to develop highly trustworthy Cloud applications within a limited source of budget and a small set of momentous type constituents needs to be identified from the Cloud based applications.

5. FTM Architecture
FTM is built to work on top of the hyper visor, straddling all the nodes and navigating the generalization layers of the Cloud to translucently tolerate failures among the progressing nodes [6]. Fig.1 depicts the architecture of FTM which can primarily be viewed as a collection of several web service constituents, each with a specific functionality.
A brief description of the functionality of all the constituents along with the underlying principle behind their enclosure in the framework is provided further in this section.

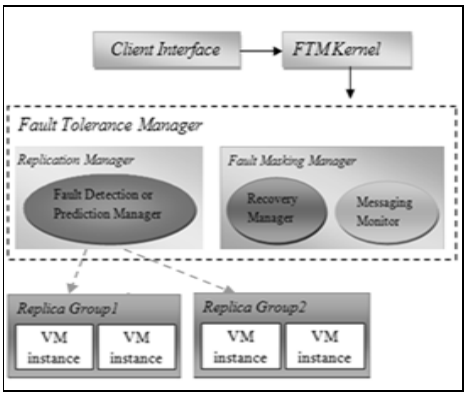


Figure 1: FTM Architecture

5.1. Client Interface

This constituent is used to obtain customer's requirements and act as an interface between the end customer and FTM. The service incantation process begins when a user requests the intermediary service provider to offer Fault Tolerance support to its Cloud based applications with a needed set of properties.

5.2. FTM Kernel

This is the main functional constituent of FTM which manages each and every trustworthy mechanism present in the proposed framework. It matches the customer's requirements and accordingly opts for the services based on the trustworthiness basis.

5.3. Replication Manager

FTM provides Fault Tolerance by reproducing customer's applications such that redundant based copy of the application is available every time when a failure occurs. The set of VM instances which is manipulated by a single implementation form of the replication services is referred as a single replica group.

5.4. Fault Detection or Prediction Manager

Fault detection building block provides the support to techniques that either detect or predict failures among the systems. Failure detection is afforded at two unique means of levels. The first level is infrastructure based centric, and provides failure detection that is global across all the nodes that are in the Cloud application, whereas the second unique level is application based centric and provides support only to detect such failures and crash among individual replicas within each replica group.

5.5. Fault Masking Manager

Fault masking manager constituent comprises of modules that are to the support methods that used to mask or hide the incidence of faults or error in the system constituent. When a fault is detected in the system constituent, this particular momentous constituent instantaneously applies the masking techniques to prevent faults from producing errors. We note that the functionality of this main constituent is difficult enough to meet the customer's high range of availability requirements.

5.6. Recovery Manager

Recovery manager entity embraces of services that support the Fault Tolerance units that recover a faulty node back to the normal mode of operation. The upshot of this failure detection and hiding type techniques on the system is completely complementary to that of the unique recovery mechanisms. By continuously keep on checking for the occasion of failure and using the recovery mode service when exceptions happen, our proposed framework increases the system's favorable lifetime and decreases the major downtime during failures.

5.7. Messaging Monitor

The main mission of the messaging monitor is that it broadens through all the major constituents of our proposed framework and offers the mere communication level infrastructure in two different formats: message level type exchange within each replica group and inter constituent level type communication form within the framework.FTM Kernel selects an suitable messaging strategy while making a Fault Tolerance type algorithm such that an autonomous service of messaging facility is available for each and every instance.

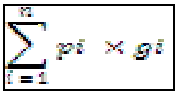
6. Optimal Fault Tolerant Strategy Selection

In the Formula given, gi will be set as 1 if the ith candidate is selected for the specific constituent else 0 will be set for this parameter. Pi, ri, ci are taken as failure probability, response time and cost of the Fault Tolerant strategy candidates, respectively. And n is considered as the number of Fault Tolerance strategy candidates for the momentous constituents and c1 and c2 are taken as the client constrictions for cost and the response time.

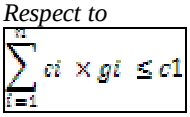
Now we first calculate the cost, response time, and the mere failure probability values of unique Fault-Tolerance type strategy candidates. It is sketched in such a way that it will select the optimal type candidate. First, the candidates that cannot meet the specified client constrictions and requirements are not included.

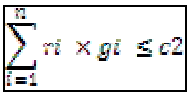
- **Formula**

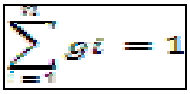
Reduce



Respect to







Next, the Fault-Tolerance candidate with gives the best failure probability value performance will be opted as the optimal Fault Tolerant strategy for the constituent. By this specific technique, the optimal Fault-Tolerance strategy, which consists of the best failure probability performance values and satisfies all the user constrictions, can be determined [9] [10].

7. Conclusion

Here, we have portrayed an approach for realizing non specific Fault Tolerance mechanisms as self regulating modules, validating Fault Tolerance possessions of each mechanism, and complementing customer's requirements with available Fault Tolerance modules to obtain a far-reaching solution with desired possessions. After finding out the momentous constituents, we propose an Optimal Fault Tolerance Strategy Selection algorithm to provide optimal Fault Tolerance mechanisms to the momentous constituents automatically, based on the user constrictions. The future work can be extended to include more user constrictions.

8. Acknowledgment

We take immense pleasure in expressing our humble note of gratitude to our project guide and coordinator, Mrs.S.Vidya, Assistant Professor from Department of Computer Science and Engineering, SNS College of Technology, Coimbatore, for her remarkable guidance and valuable suggestions, which helped us in completing the paper effectively.

9. References

1. Ravi Jhawar, Vincenzo Piuri, and Marco Santambrogio, "Fault Tolerance Management in Cloud Computing: A System-Level Perspective", IEEE systems journal, vol. 7, no. 2, June 2013.
2. Wenbing Zhao, IEEE, Melliar-Smith P. M. and Moser L. E., IEEE, "Fault Tolerance Middleware for Cloud Computing", 2010 IEEE 3rd International Conference on Cloud Computing.
3. Prasenjit Kumar Patra, Harshpreet Singh, Gurpreet Singh, "Fault Tolerance Techniques and Comparative Implementation in Cloud Computing", IJCA, February 2013.
4. Anjali D.Meshram, A.S. Sambare, S.D.Sade, "Fault Tolerance Model for Reliable Cloud Computing", IJRITCC July 2013.
5. Sheheryar Malik, IEEE, Fabrice Huet, IEEE, "Adaptive Fault Tolerance in Real Time Cloud Computing", 2011 IEEE World Congress on Services.
6. Ravi Jhawar, Vincenzo Piuri, and Marco Santambrogio, "A Comprehensive Conceptual System- Level Approach to Fault Tolerance in CloudComputing", 2012 IEEE International Systems Conference.
7. Amazon elastic compute Cloud.
8. Alain Tchana, Laurent Broto, Daniel Hagimont "Approaches to Cloud Computing Fault Tolerance", IEEE 2012 International Conference.
9. Zibin Zheng and Michael R. Lyu, "Optimal Fault Tolerance Strategy Selection for Web Services", IJWSR 2010.
10. Zibin Zheng and Michael R. Lyu, "A QoS Aware Fault Tolerant Middleware for Dependable Service Composition" IEEE International Conference 2009.
11. N-Version Programming www.hillside.net/plop/2009/Process/N-Version Programming.pdf.
12. Vidya, S., K. Vani, and D. Kavinpriya. "Secured Personal Health Records Transactions Using Homomorphic Encryption In Cloud Computing." International Journal of Engineering 1.10 (2012)