International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 8 (2015) pp. 19401-19409

http://www.ripublication.com

Reactive Fault Tolerance In Cloud Computing

Shyamala Gowri S[#1] and Kiruthika B [#2]

 # [1]Student, Sathyabama University ,
Jeppiaar Nagar, Shollinganallur, Chennai - 600 119, Tamil Nadu, India.
shamlovescello@gmail.com
kiruthika2balu@gmail.com

D.Kaavya[#3],

 # [3]Lecturer, Sathyabama University ,
Jeppiaar Nagar, Shollinganallur, Chennai - 600 119, Tamil Nadu, India.
d.kaavya89@gmail.com

Abstract

Fault tolerance is one of the major issues in guaranteeing the provisioning and reliability of system services and as well as execution of an application. To reduce the failures that occur on the device and application, they need to be handled and taken up after their occurrence. There are many techniques to deal this fault tolerance. These techniques are used to take an appropriate action after failures occur. This particular paper discusses the existing fault tolerance systems in cloud computing based on the policies, tools and research challenges. Visualization of cloud architecture has been proposed here in this paper. In the proposed system autonomic fault tolerance has been implemented. The derived results demonstrate that the proposed system can deal with various software faults for server applications in a cloud virtualized environment. That is,a system as a whole is not hanged due to problems either in the hardware or the software at any case. An example in this field is a motor vehicle designed so it will continue to be drivable even if one of the tires is punctured. A structure retains its integrity in the presence of damage because of fatigue, corrosion, manufacturing or impact.With the scope of an individual system, fault tolerance can be targeted by considering exceptional conditions and building the system to cope with them, and, in other words, aiming for self-stabilization so that the system converges towards an erroneous state.

Keyword: Cloud Computing; Virtual Machine; Fault, Tolerance; Replication
Fault Tolerance on Large Scale Systems using Adaptive Process Replication

Introduction

Cloud computing involves deploying groups of remote servers and software networks that allow centralized data storage and online access to computer services or resources. Cloud domains can be classified as public, private or hybrid. Here, we have illustrated the cloud environment with the help of a tool called UltiDev. Fault-tolerance is the phenomenon that enables a system or platform to continue operating properly in the event of the failure of (or one or more faults) some of its components. Whenever a fault is found, the technique provides mechanisms to the running application system in order to prevent the occurrence of failure. The major advantages of implementing fault tolerance in cloud computing include failure recovery, reduced cost, greater performance metrics and so on. This paper gives a better perspective of fault tolerance challenges with scenarios and identifies techniques used for the same. When several instances of an application are running on many virtual machines and one of the servers goes down, then there is a need to implement the concept of fault tolerance which can handle these types of failures across. To elaborate this issue, cloud's virtualized system architecture has been given and implemented using a proxy medium. The considered architecture also has been validated eventually through experimental results along with the effects of implementation.

Reactive Fault Tolerance:

Reactive fault tolerance policy reduces the effect of faults on an application execution when the failure effectively occurs. There are various techniques that are based on such policies like Checkpoint/Restart, Replay, Retry, Software Rejenuvation, Masking etc., In our proposed approach, Hybrid Particle Swarm Optimization (HPSO) algorithm and any cast technique are being used in order with the conjunction of a middleware. We have adopted a reactive fault management strategy for centralized and distributed environments especially in the cloud. The proposed strategy helps in identifying the root cause of failures or faults and resolves such cognitive problems. In our strategy, there is a minimized computation logic and communication that supports higher reliability. In case of any effect of Denial of Service/Distributed Denial of Service DOS or DDOS attacks nearest to the source of the attack, our methodology helps in reaching higher security. Significance of performance improvement in this is achieved by using any cast before HPSO. The selection and usage of more reliable nodes results in less and optimal overhead of check pointing.

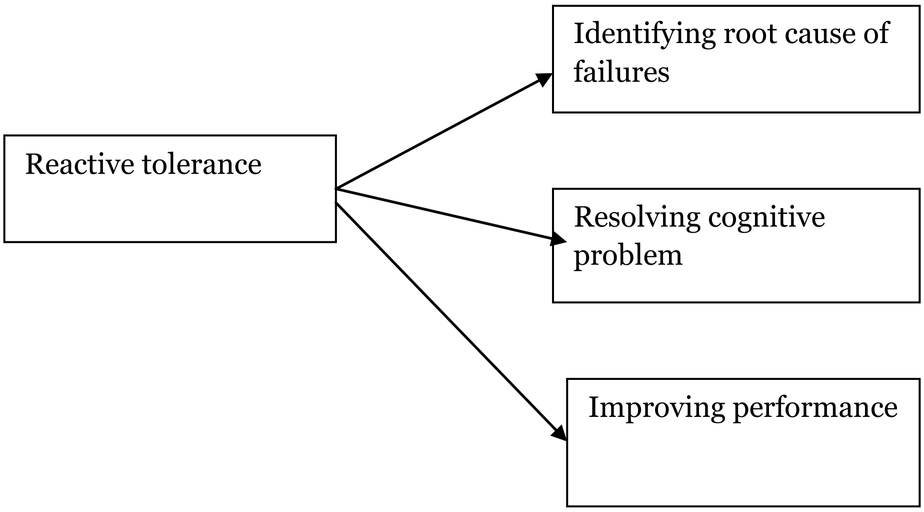

Figure 1: Reactive Tolerance

Challenges in Implementing Fault Tolerance:

Providing fault tolerance requires keen consideration and analysis due to their complexity, inter-dependency and the below given reasons,

- There is a necessity for implementing autonomic fault tolerance technique for several instances of an application that is running on several virtual machines.
- Different technologies from various vendors of cloud platform need to be integrated for establishing a reliable system.
- A new approach has to be developed so as to integrate the fault tolerance techniques with an existing workflow scheduling algorithms as required.
- A criterion based method can be deployed in cloud environment for evaluating the performance of fault tolerant component in comparison with similar ones.
- To ensure high reliability and vacancy, multiple cloud computing providers with autonomous software stacks should be used.

RFS Implementation:

In the phase of service requirement, the service has to fulfill the consumer details, the technical and also functional specifications. While defining the service requirements, the consumer specifies the nonfunctional attributes such as characteristics of the human agent providing the service, constraints and importance on data standard, and required encryption policies for the service. Service yielding details like certifications needed, standards to be followed, and so on, are also identified by then. The technical specifications include the hardware entities, softwares, application set and language policies to which a service should stick to. Once the consumers or actors have identified and classified their service needs, they issue a Request for Service (RFS). This particular RFS can be generated in a machine readable format using Semantic Web technologies. If the user is not satisfied with the services he/she has discovered, they are free to change the requirements ,say, by increasing the cost constraint or policies and restart the discovery phase with a new RFS feed. We assume that the user requirements will change once the user begins to consume the services and hence we show a link between consumption and the requirements phase.

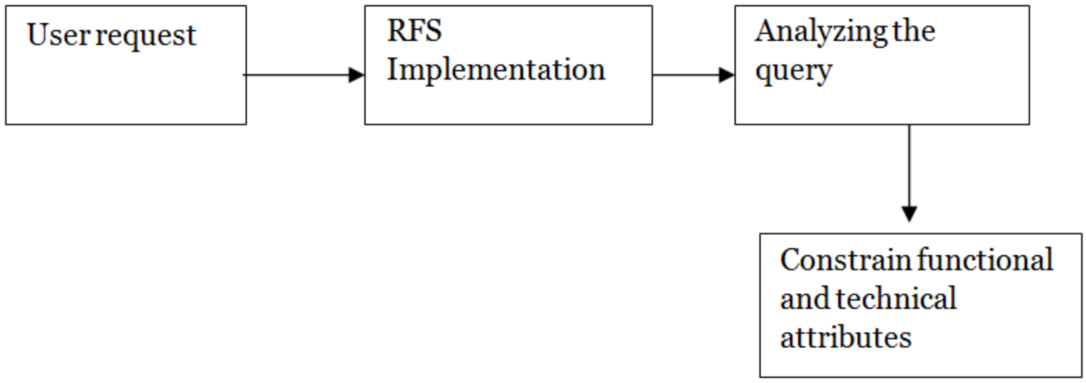

Figure 2: Request For Service based on user query

Service Selection:
In this selection phase, services are selected by analyzing the user query in the RFS with respect to the service descriptions. The selected service gets constrained by functional and technical attributes defined, and also by the budgetary, security and agent policies of the consumer. An organization can turnout the RFS to a limited set of providers. More readily, it can search for all possible vendors available on the internet. This selection of services from RFS can provide user requested details about different social networks which can be employed.

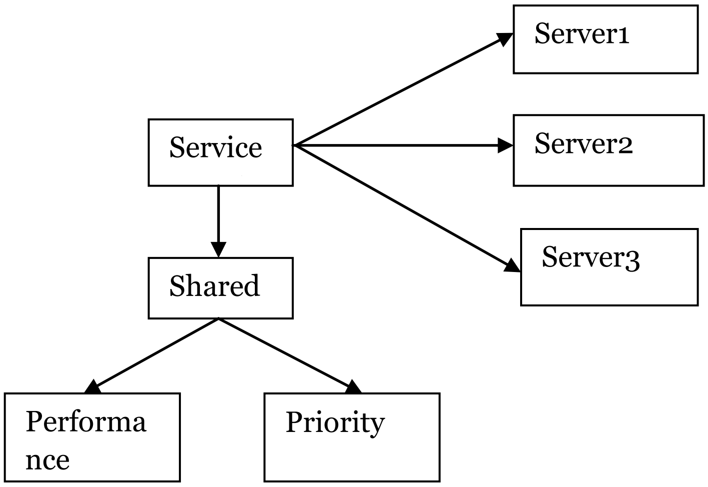

Figure 3: Selection of Service from Cloud

Service Composition:

It is an aggregate and collection of services composed in order to automate a specific task or a business process. To authorize as a composition, a minimum of two participating services plus one composition initiator must to be present. Otherwise, the service interaction will only represent a point to point exchange.

Web Service is the most peculiar implementation of service-oriented architectures that have brought few challenging research concerns. One of them is the composition, i.e., the potential to continuously build a composite website .The selection of a Web service, for each activity of the workflow model, meeting the user requirements, is still a worthy challenge. In fact, the selection of one Web service from a set of them that fulfill some performances is a critical task, basically depending on the combined evaluation of Quality of Service. However, the conventional QoS-aware composition approaches do not consider the transactional limitations during the process of composition. Here in this paper, we are clubbing the service composition at one stage with fault occurrence due to Denial of Service (DoS). We propose a selection algorithm that satisfies user preferences being expressed as weights over QoS criteria and as risk levels defining semantically the process requirements. This is called Risk Driven Selection Algorithm.

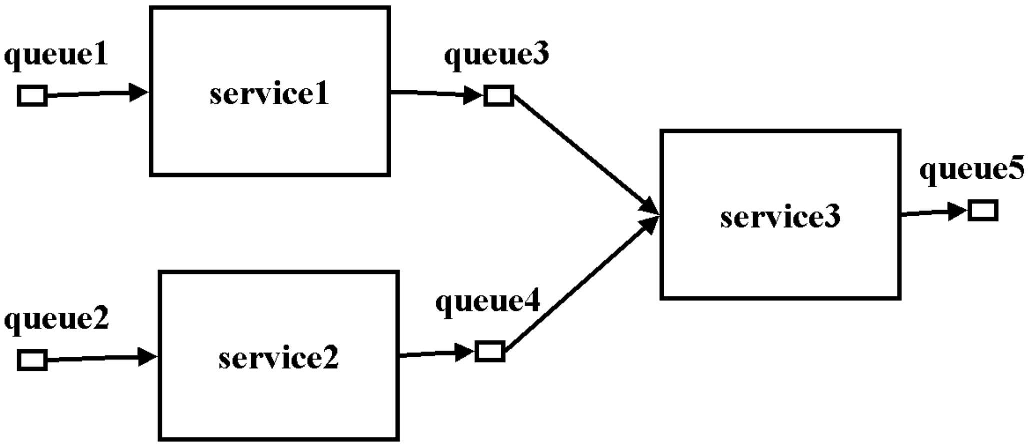

Figure 4: Service Composition based on risk selection

Differentiated Services:

Once after the servers are clustered, tasks are allocated that it can be assigned to each server and calculate the performance and as well as priority. Each machine can host multiple applications. The applications preserve their current state in the backend storage servers. It is important to note that the applications themselves are stateless in order to be replicated safely.

Distributional Classes:

The system under consideration contains n number of servers that render service in the order of task request arrivals as per First Come First Serve (FCFS).The capacity of system is m þ r which means the size of the buffer for incoming request is equal to r. As the population size of normal cloud center is relatively high, the probability that a given user will request for service is comparatively small, and hence the arrival process can be modeled as an efficient process.

Masking Process:

Masking process in fault-tolerance mechanism guarantees that the programs satisfy their specification in the presence of faults continuously. By means of contrast, non-masking of fault-tolerance is vice versa: it merely guarantees that when faults stop occurring, the program executions converge to different states from where programs satisfy their specification. This is the transformed state. Here, we present a component based method for the design of masking fault-tolerant programs. In this, components are added to a fault-intolerant program in a stepwise or sequential manner so as to transform the fault-intolerant program into a non-masking fault-tolerant one and then, to enhance the fault-tolerance from non-masking to masking. We illustrate the method by illustrating programs for agreement in the presence of specific faults, data migration in the presence of information loss, triple modular redundancy in the presence of input variation, and mutual exclusion in the presence of process fail-stops with respect the operating system. The system provides a recovery mechanism after the occurrence of failure. These examples also serve to demonstrate that the method accommodates and includes a variety of fault-classes. It provides alternative platforms for programs usually designed with extensive design models and further offers the capacity for better masking fault-tolerant programs.

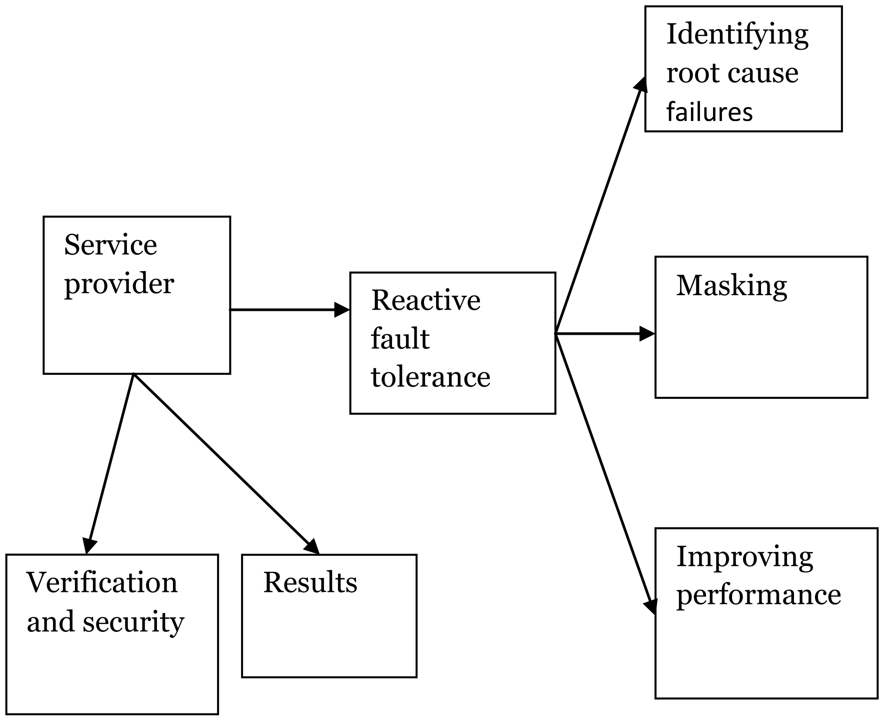

Figure 5: Masking Process for Fault Rectification

Algorithms Used:

- Paxos Algorithm
- Round Robin Algorithm

Paxos Algorithm:

In this algorithm, three agents are being used.

- They are proposers, acceptors and learners respectively. The proposers are those that propose values. To differentiate different proposals, each proposal must carry a unique and increasing proposal number v.
- The acceptors are the ones who accept or reject the proposals. If the majority of the acceptors accept a proposal with the value d, then it is said that the value d has been chosen by the group of acceptors.
- The learners are those who must find if a value has been chosen or not.
- The Paxos algorithm works in two phases. In phase I, a proposer sends a prepare request with a proposal number v to the acceptors.
- In response to the prepare request, the acceptor sends the proposer a promise or verdict that it will not accept any more proposals that are numbered less than v, and the highest-numbered proposal, if any, that it has accepted, is not a higher-numbered proposal.

- During phase II, the proposer sends an accept request to the acceptors with the proposal number v and a key value d, provided that it has collected responses correspondingly. The value d is the one in the highest numbered proposal among the responses to the prepare request, or any random value selected by the proposer if none of the acceptors has accepted any previous proposal.
- An acceptor accepts the accept request with v and d provided that it has not responded to a prepare request with a higher proposal number as a condition. After a value has been chosen, there must be a method for the learners to find this out.
- A simple way to achieve this is for the proposer to publish the chosen value to the learners at once in the beginning.

Round Robin Algorithm:

Here, in this procedure, each process gets placed in a run-queue and every process is allocated a service quantum of time which is commonly set to 10 milliseconds.

- The processes that demand less time run without being interrupted by any external units. Similarly, processes that exceed the service quantum are interrupted and returned to the back of the run-queue in order to wait for further processing of the same.
- Round Robin algorithm favors short process demands in common. Consequently, it is biased in benefit of a greedy user who runs many short-demand processes with the cloud system.
- The operating system is mainly based on the environment that the developer chooses.

Conclusion:

Fault tolerance scenario is concerned with all the mechanisms necessary to enable a system that tolerates software faults in the system after its development. This paper illustrated about the fault tolerance techniques including its research challenges, methods used for implementing fault tolerance in cloud computing. Cloud system architecture is also proposed based on virtualization. Reactive fault tolerance is implemented with various software faults for server applications in a cloud environment. When one of the working servers goes down unexpectedly, connection will naturally be redirected to the other server while accessing the service. The experimental results obtained justify the system fault tolerance.

Acknowledgment:

We would like to thank the management of Sathyabama University for their support in publishing this paper.

References:

[1] Jasbir Kaur and Supriya Kinger- Efficient Algorithm for Fault Tolerance in Cloud Computing in the year 2014.
[2] Anju Bala and Inderveer Chana-Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing in the year 2012.
[3] Antonina Litvinova, Christian Engelmann and Stephen L. Scott," A Proactive Fault Tolerance Framework for High Performance Computing", 2009.
[4] Golam Moktader Nayeem, Mohammad Jahangir Alam," Analysis of Different Software Fault Tolerance Techniques", 2006.
[5] Steven Y. Ko, Imranul Hoque, Brian Cho and Indranil Gupta, "On Availability of Intermediate Data in Cloud Computations", 2010.