ISSN:2249-5789

FAULT TOLEREANE TECHNIQUES AND ALGORITHMS IN CLOUD COMPUTING

Dr. Lakshmi Prasad Saikia
Professor & Head, Dept. of Computer Sc. & Engineering
Assam down town University, Guwahati, India
Email: lp_saikia@yahoo.co.in

Yumnam Langlen Devi
B. Tech. Student, Dept. of Computer Sc. & Engg.
Assam down Town University, Guwahati, India
E-mail: langlentombi@gmail.com

Abstract

Cloud computing provide many service to the user. Now a day the user level is highly increased to utilize the services in cloud computing. In cloud computing the major problem area is fault tolerance. Fault tolerance is a major concern to guarantee availability and reliability of critical services as well as application execution. In order to minimize failure impact on the system and application execution, failures should be anticipated and handle. Fault tolerance techniques are used to predict these failures and take an appropriate action before or after failures occur.

1. INTRODUCTION

Cloud computing is sharing of resources on a larger scale which is cost effective and location independent. Resources on the cloud can be deployed by the vendor, and used by the client. It also shares necessary software's and on-demand tools for various IT Industries. Amazon is the first company to look into the growing importance of Cloud computing very seriously followed by Google and IBM. It is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). Cloud computing emerges as a new computing paradigm which aims to provide reliable, and quality of service guaranteed computing lively environments for end-users. The increasing demand for flexibility in obtaining and releasing computing resources in a cost-effective manner has resulted in a wide adoption of the Cloud computing paradigm.

Fault tolerance computing is one that can continue to correctly perform its task in the presence of hardware failures and/or software or to operate satisfactorily in the presence of faults. Fault tolerance bear-on with all the inevitably techniques to enable robustness and dependability .The main benefits of implementing fault tolerance in cloud computing include failure recovery, lower cost, improved performance metrics. Robustness leads to the property to providing of a correct service in an adverse situation arising due to an uncertain system environment. Dependability is related to some QoS (Quality of Services) aspects provided by the system, it includes the attributes like reliability and availability.

2. CLOUD SYSTEMS

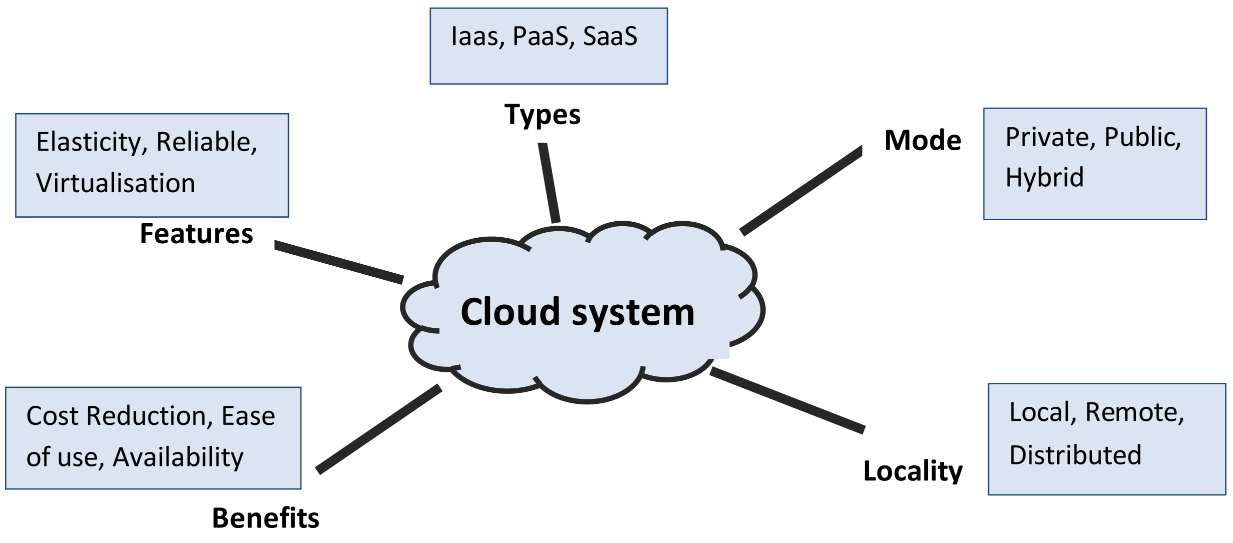
Iaas, PaaS, SaaS
Types
Elasticity, Reliable, Virtualisation
Features
Mode
Private, Public, Hybrid
Cloud system
Cost Reduction, Ease of use, Availability
Benefits
Locality
Local, Remote, Distributed

Fig. 1: - Cloud system

2.1. TYPES OF CLOUDS

Infrastructure as a Service (IaaS) also referred to as Resource Clouds, provide (managed and scalable) resources as services to the user; In other words, they basically provide enhanced virtualisation capabilities. Accordingly, different resources may be provided via a service interface: Data & Storage Clouds deal with reliable access to data of potentially dynamic size, weighing resource usage with access requirements and / or quality definition.

Examples: Amazon S3, SQL Azure.

Platform as a Service (PaaS), provide computational resources via a platform upon which applications and services can be developed and hosted. PaaS typically makes use of dedicated APIs to control the behaviour of a server hosting engine which executes and replicates the execution according to user requests (e.g. access rate). As each provider exposes his / her own API according to the respective key capabilities, applications developed for one specific cloud provider cannot be moved to another cloud host.

Examples: Google App Engine, Windows Azure (Platform).

Software as a Service (SaaS), also sometimes referred to as Service or Application Clouds are offering implementations of specific business functions and business processes that are provided with specific cloud capabilities, i.e. they provide applications / services using a cloud infrastructure or platform, rather than providing cloud features them. Often, kind of standard application software functionality is offered within a cloud.

Examples: Google Docs, Salesforce CRM, SAP Business by Design.

2.2. CLOUD USAGE

Private Clouds are typically owned by the respective enterprise and / or leased. Functionalities are not directly exposed to the customer, though in some cases services with cloud enhanced features may be offered – this is similar to (Cloud) Software as a Service from the customer point of view.

Example: eBay.

Public Clouds enterprises may use cloud functionality from others, respectively offer their own services to users outside of the company. Providing the user with the actual capability to exploit the cloud features for his / her own purposes also allows other enterprises to outsource their services to such cloud providers, thus reducing costs and effort to build up their own infrastructure. As noted in the context of cloud types, the scope of functionalities thereby may differ.

Example: Amazon, Google Apps, Windows Azure.

Hybrid Cloud, though public clouds allow enterprises to outsource parts of their infrastructure to cloud providers, they at the same time would lose control over the resources and the distribution management of code and data. In some cases, this is not desired by the respective enterprise. Hybrid clouds consist of a mixed employment of private and public cloud infrastructures so as to achieve a maximum of cost reduction through outsourcing whilst maintaining the desired degree of control over e.g. sensitive data by employing local private clouds.

2.3. CHARACTERISTICS / CAPABILITIES OF CLOUDS

Elasticity is an essential core feature of cloud systems and circumscribes the capability of the underlying infrastructure to adapt to changing, potentially non-functional requirements, for example amount and size of data supported by an application, number of concurrent users etc.

Reliability is essential for all cloud systems – in order to support today’s data centre-type applications in a cloud, reliability is considered one of the main features to exploit cloud capabilities. Reliability denotes the capability to ensure constant operation of the system without disruption, i.e. no loss of data, no code reset during execution, etc.

Quality of Service support is a relevant capability that is essential in many use cases where specific requirements have to be met by the outsourced services and / or resources.

Availability of services and data is an essential capability of cloud systems and was actually one of the core aspects to give rise to clouds in the first instance. It lies in the ability to introduce redundancy for services and data so failures can be masked transparently. Fault tolerance also requires the ability to introduce new redundancy (e.g. previously failed or fresh nodes) in an online manner non-intrusively (without a significant performance penalty).

Cost reduction is one of the first concerns to build up a cloud system that can adapt to changing consumer behaviour and reduce cost for infrastructure maintenance and acquisition. Notably, setting up a cloud system typically entails additional costs – be it by adapting the business logic to the cloud host specific interfaces or by enhancing the local infrastructure to be “cloud ready”.

3. TYPES OF FAULTS

These faults can be classified on several factors such as:

Network fault: A Fault occur in a network due to network partition, Packet Loss, Packet corruption, destination failure, link failure, etc.

Physical faults: This Fault can occur in hardware like fault in CPUs, Fault in memory, Fault in storage, etc.

Media faults: Fault occurs due to media head crashes.

Processor faults: fault occurs in processor due to operating system crashes, etc.

Process faults: A fault which occurs due to shortage of resource, software bugs, etc.

Service expiry fault: The service time of a resource may expire while application is using it.

A fault can be categorized on the basis of computing resources and time. A failure occurs during computation on system resources can be classified as: omission failure, timing failure, response failure, and crash failure. Fault may be:

Permanent: These failures occur by accidentally cutting a wire, power breakdowns and so on. It is easy to reproduce these failures. These failures can cause major disruptions and some part of the system may not be functioning as desired.

Intermittent: These are the failures appears occasionally. Mostly these failures are ignored while testing the system and only appear when the system goes into operation. Therefore, it is hard to predict the extent of damage these failures can bring to the system.

Transient: These failures are caused by some inherent fault in the system. However, these failures are corrected by retrying roll back the system to previous state such as restarting software or resending a message. These failures are very common in computer systems.

3.1. EXISTING FAULT TOLERANCE TECHNIQUES IN CLOUD COMPUTING

Various fault tolerance techniques are currently prevalent in clouds:-

Check pointing–It is an efficient task level fault tolerance technique for long running and big applications .In this scenario after doing every change in system a check pointing is done. When a task fails, rather than from the beginning it is allowed to be restarted that job from the recently checked pointed state.

Job Migration –Some time it happened that due to some reason a job can- not be completely executed on a particular machine. At the time of failure of any task, task can be migrated to another machine. Using HA-Proxy job migration can be implemented.

Replication-Replication means copy. Various tasks are replicated and they are run on different resources, for the successful execution and for getting the desired result. Using tools like HA-Proxy, Hadoop and AmazonEc2 replication can be implemented.

Self- Healing- A big task can divided into parts .This Multiplication is done for better performance. When various instances of an application are running on various virtual machines, it automatically handles failure of application instances.

Safety-bag checks: In this case the blocking of commands is done which are not meeting the safety properties.

S-Guard- It is less turbulent to normal stream processing. S-Guard is based on rollback recovery. S-Guard can be implemented in HADOOP, Amazon EC2.

Retry- In this case we implement a task again and gain. It is the simplest technique that retries the failed task on the same resource.

Task Resubmission- A job may fail now whenever a failed task is detected, In this case at runtime the task is resubmitted either to the same or to a different resource for execution.

Timing check: This is done by watch dog. This is a supervision technique with time of critical function.

Rescue workflow- This technique allows the workflow to persist until it becomes unimaginable to move forward without catering the failed task.

Software Rejuvenation-It is a technique that designs the system for periodic reboots. It restarts the system with clean state and helps to fresh start.

Pre-emptive Migration- Pre-emptive Migration count on a feedback-loop control mechanism. The application is constantly monitored and analysed.

Masking: After employment of error recovery the new state needs to be identified as a transformed state. Now if this process applied systematically even in the absence of effective error provide the user error masking.

Reconfiguration: In this procedure we eliminate the faulty component from the system.

4. RELATED WORKS

Since about 1970, the field of fault tolerance computing has been rapidly developing. In journals like IEEE transaction on Computers, IEEE Transaction on Reliability, IEEE Computers etc., started publishing papers and other related works. In 1971-1975, introducing new Microprocessor, Intel 4004, 8080 and in 1975, Tandem computer tailor its Tandem-16, the first fault tolerance computer for online transaction processing.

In 1999, Felix C. Gartner had tried to use a formal approach to structure the area of fault-tolerant distributed computing, survey fundamental methodologies, and discuss their relations. The advantages of a formal approach, however, also lie in the fact that it reveals the inherent limitations of fault tolerance methodologies and their interactions with system models. Stating the impossibility of unconditional dependability is as important as trying to build systems that are increasingly reliable.

In 2004, Lee Pike et al. introduced four fundamental abstractions in the domain of fault tolerant distributed systems. *Message Abstractions address* the correctness of individual messages sent and received. *Fault Abstractions address* the kinds of faults possible as well as their effects in the system. *Fault-Masking Abstractions address* the kinds of local computations processes make to mask faults. Finally, *Communication Abstractions address* the kinds of data communicated and the properties required for communication to succeed in the presence of faults.

In 2011, Arvind Kumar et. al., Explore various reliable fault detection and fault tolerance methods. Their main focus is on types of fault occurring in the system, fault detection techniques and the recovery techniques used. This paper provides how these methods are applied to detect and tolerate faults from various Real Time Distributed Systems. Most Real time systems function with very high availability even under hardware fault conditions. The most useful hardware fault tolerance techniques are redundancy and load sharing. For tolerating any fault from the system first they require to detect the fault occurred in the system and then isolating it to the appropriate unit as quickly as possible. The main detection mechanisms are: Sanity Monitoring, Watchdog Monitoring, Protocol Faults, In-service Diagnostics, and Transient Leaky Bucket Counters. If a unit is really faulty, many fault triggers will be generated for that unit. The main objective of fault isolation is to correlate the fault triggers and identify the faulty unit.

In 2011, Zhang et. al. Proposed a Byzantine fault tolerance framework for building reliable system in voluntary resource cloud infrastructure". The input of BFT Cloud is a sequence of requests with specified

QoS requirements (e.g., preferences on price, capability, bandwidth, workload, response latency, failure probability, etc.) sent by the cloud module. The output of BFT Cloud is a sequence of committed responses corresponding to the requests. The responsibilities of primary include: accepting requests from the cloud module, selecting appropriate replicas to form a BFT group, forwarding the request to all replicas, and replacing faulty replicas with newly selected nodes. Since failures happened on primary will greatly decrease the overall performance of a BFT group. The cloud module first forms a request sequence and sends the sequence of requests to the primary. The primary will order the requests and forward the ordered requests to all the BFT group members. Each member of the BFT group will execute the sequence of requests and send the corresponding responses back to the cloud module. The cloud module collects all the received responses from the BFT group members and makes a judgement on the consistence of responses. An action strategy will be chose according to the consistence of responses. When the primary is faulty, a primary updating procedure will be triggered in the Request Execution phase.

In 2012, Anju Bala and Indrerveer Chana proposed a cloud virtualized system architecture. In the proposed system autonomic fault tolerance has been implemented. If any one of the servers breaks down, system should automatically redirect user requests to the backup server. The server virtualized system consists of VMs (server 1 and server 2) on which an Ubuntu 10.04 OS and database application are running. Server 2 is a backup sever in case of failure. HAProxy is configured on the third virtual machine to be used for fault tolerance. The availability of the servers is continuously monitored by HAProxy statistics tool on a fault tolerant server. HAProxy is running on web server to handle requests from web. When one of the servers goes down unexpectedly, connection will automatically be redirected to the other server.

In 2013, Patra and Singh approached fault taxonomy and need of fault tolerance in cloud computing. Various proposed models for fault tolerance are discussed and compared on the basis of Metrics for fault tolerance in cloud. In the present scenario, there are number of fault tolerance models which provide different fault tolerance mechanisms to enhance the system. But still there are number of challenges which need some concern for every frame work or model.

In 2013, N. Chandrakala and P. Sivaprakasam purposed a load balancing algorithm for virtual machine. The VM load balancing algorithm is used to balance the load in the cloud pool. This algorithm check the CPU utilization depends upon the request. In proposed method the dynamic cloud computing environment is used. The intermediate node is used to monitor the load of each VM in the cloud pool. In this approach the user can send the request to the intermediate node. It is responsible for transfer the client request to the cloud. Here, the load is considered as in terms of CPU load with the amount of memory used, delay or Network load. In proposed method, took two VM for test. They checked the CPU usage and the memory usage of the two VM and identify the reliable VM which contains fewer loads and high memory can process the client request. The balancing section is responsible for determining where virtual machines will be instantiated. In this algorithm the load balancer node check the CPU utilization if the CPU is less than 75% utilization the request accepts otherwise VM load balancing Algorithm instantiates a new virtual machine on the compute node with the lowest utilization number. The algorithm is to identify the reliable VM and process the client request. For that the algorithm cerate cloud pool. The cloud pool contains the VM.

In 2013, Anjali D. Meshramet et al. designed a system that deals with the fault tolerance mechanism In this system , a model name fault tolerance model for cloud (FTMC) model is based upon reliability assessment of computing nodes known as a virtual machines(VM) in cloud environment and fault tolerance of real time applications running on those VMs. A virtual machine is selected for computation on the basis of its reliability and can be removed, if does not perform well for real time applications. In this scheme, "N" virtual machine; which run the "N" variant algorithms. Algorithm "X1" runs on "Virtual machine-1", "X2" a run on "Virtual machine-2", up till "Xm", which runs on "Virtual machine -m". Then there is ACCEPTER which is responsible for the verification of output result of each node. The outputs are then passed to TIMER which checks the timing of each result. On the basis of the timing the RELIABILITY ASSESSOR calculates and reassigns the reliability of each module. Then all the results are forwarded to DECISION MAKER which selects the output on the basis of best reliability. The output of a node with highest reliability is selected. DECISION MAKER also request to some responsible module (resource manager or scheduler) to remove one node with minimum reliability and add a new node.

In 2013, Ravi et. al. introduced an innovative, system-level, modular perspective on creating and managing fault tolerance in Clouds. They proposed a comprehensive high-level approach to shading the implementation details of the fault tolerance techniques to application developers and users by means of a dedicated service layer. Furthermore, they present a scheme that: (i) delivers a comprehensive fault tolerance solution to user's applications by combining selected fault tolerance mechanisms, and (ii) ascertains the properties of a fault tolerance solution by means of runtime monitoring. Based on the proposed approach, designed a framework that easily integrates with the existing Cloud infrastructure and facilitates a third party in offering fault tolerance as a service. A client engages with the service provider to obtain fault tolerance support for its applications. The goal of the service provider is to create a fault tolerance solution based on the client's requirements. The service provider must maintain a consistent view of all computing resources in the Cloud to efficiently allocate resources during each client request and to avoid over provisioning during failures.

5. CONCLUSION

There are number of fault tolerance models which provide different fault tolerance mechanism to enhance the system. There are some drawbacks that can't be fulfilling the all aspects of faults. A possibility to overcome the drawbacks of all previous models and try to make a compact model which will cover maximum fault tolerance aspect. Fault tolerance methods come into play the moment a fault enters the system. A lot of techniques have been developed for achieving fault tolerance in software. The application of all of these techniques is relatively new to the area of fault tolerance. Furthermore, each technique will need to be tailored to particular applications.

REFERENCES

[1] Felix C. Gärtner, "Fundamentals of Fault-Tolerant Distributed Computing in Asynchronous Environments", ACM Computing Surveys, Vol. 31, No. 1, March 1999.

[2] Lee Pike, Jeffrey Maddalon, Paul Miner, and Alfons Geser, "Abstractions for Fault-Tolerant Distributed System Verification", In Theorem Proving in Higher Order Logics (TPHOLs), volume 3223 of Lecture Notes in

Computer Science, pages 257–270, Springer, 2004.

[3] Yilei Zhang, Zibin Zhengand and Michael R. Lyu, "BFTCloud: A Byzantine Fault Tolerance Framework for Voluntary-Resource Cloud Computing", 4th International Conference on Cloud Computing, IEEE, 2011.

[4] Arvind Kumar, Rama Shankar Yadav, Ran vijay and Anjali Jain "Fault Tolerance in Real Time DistributedSystem", International Journal on Computer Science and Engineering (IJCSE),Vol. 3,ISSN: 0975-3397, No. 2 ,Feb 2011.

[5] BanM. Khammas, "Design a Fault Tolerance for Real Time Distributed System", Al-Khwarizmi Engineering Journal, Vol. 8, No. 1, PP11 -17, 2012.

[6] AnjuBala, InderveerChana, "Fault Tolerance-Challenges, Techniques and Implementation in Cloud Computing", International Journal of Computer Science (IJCSI) Issues, Vol. 9, Issue 1, No 1, January 2012.

[7] Ravi Jhawar, Vincenzo Piuri and Marco Santambrogio, Member of IEEE, "Fault Tolerance Management in Cloud Computing: A System-Level Perspective ", IEEE, 2012 .

[8] Sourabh Dave and AbhishekRaghuvanshi, "Fault Tolerance Techniques in Distributed System", International Journal of Engineering Innovation & Research Vol. 1, Issue 2, ISSN: 2277 – 5668, 2012.

[9] Prasenjit Kumar Patra ,Harshpreet Singh &Gurpreet Singh, "Fault Tolerance Techniques and Comparative Implementation in Cloud Computing" International Journal of Computer Applications , Vol. 64, No.14, February 2013.

[10] Anjali D. Meshram, A.S.Sambare and S. D. Zade , "Fault Tolerance Model for Reliable Cloud Computing ", International Journal on Recent and Innovation Trends in Computing and Communication, ISSN 2321 – 8169, Vol. 1, Issue: 7, July 2013.

[11] N. Chandrakala and Dr. P. Sivaprakasam, " Analysis of Fault Tolerance Approaches in Dynamic Cloud Computing ", International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X , Vol. 3, Issue 2, February 2013.

[12] Krish Jamsa, "Cloud Computing", First India Edition, Jones and Bartlett India Pvt. Ltd., 978-93-80853-77-2, 2013.