Reactive fault tolerance strategies

Abstract

Cloud is the buzzword among computational technologies. It has brought a paradigm shift in a way computing is done and data is stored. This cost-effective means of technology has attracted a lot of people towards it and companies are embracing cloud to reduce their operational costs. As grows the popularity so will the challenges. One of the foremost challenges is Fault Tolerance. Fault Tolerance ensures the availability, reliability, and performance of the cloud applications. This paper is mainly focused on the Reactive fault tolerance strategies. Firstly, the paper outlines various faults, errors, and failures in the Cloud Computing scenario. Then, various prevalent reactive fault tolerance strategies are discussed. Lastly, a comparative analysis is done to better understand the application of the discussed strategies.

I. Introduction

Cloud-Computing is gaining traction due

II. Faults, Errors, and Failures

2. 1 Faults.

Fault is the cause of the system or a component in the system to fail. Faults induce errors into system which hinders the ability of any system to perform as expected and give desired results. An erroneous system ultimately leads towards failure. Fault tolerance is the ability of the system to keep going in presence of one or more faults but with decaying performance. We must thoroughly classify and analyze various kinds of faults, errors, and failures to

come up with sound Fault-Tolerance Strategies. Faults in Cloud Computing environment can be classified as follows.

Aging related fault

As time passes, these faults show up into the system. These can be further categorized into two types namely Software based aging and Hardware based aging. Once the software starts execution, there is an accumulation of software bugs in the system. Furthermore, the decaying performance of the system hardware makes the system incapable to perform to its requirements.

Omission fault

This kind of faults occur when the resources in the system dry up and eventually the ongoing processes end up falling short of the resources in terms of storage capacity and computing power. Omission faults are mainly of two types i. e. Denial of Service, where the attacker tries to make the resources unavailable to its intended users by overwhelming the system with too many superfluous requests. The other type is Disk Space Full, in which the amount of free space required by the applications is no longer available, this leads to node failure?

Response faults.

Response faults occur when the server gives an incorrect response to a query made by the user. This is further classified into 3 types. Value Faults-If faults at an application level or at lower level in the system are not managed properly, this can cause the individual application or the processor to emit an

incorrect value. Byzantine faults- This fault attributes to the erratic behavior of the processor when it gets corrupted. The processor has not stopped working but the results are not predictable. State transition faults- When systems change their states, this kind of fault surfaces.

Timing Faults.

Synchronization is a key factor when it comes to execution of tasks in a distributed computing ecosystem. There should be time constraints for communication and execution of tasks by the processor. Faults which arise due to poor synchronization are called Timing faults. If the communication or the task execution begins early, then it is called Early fault. If the processor takes a lot of time to execute the tasks and this results in undesirable delay in the communication, then it is called Late fault.

Interaction Faults.

As the number of services grow in the system along with its complexity, the interaction between the services also increases. This may cause faults which occur due to Policy and Security incompatibilities. Various service providers have different policies and different security protocols.

Life Cycle Faults.

The service time of an application may expire when a user is trying to use that application. User cannot further access it unless the service becomes active again. This is called as Life Cycle fault or Service expiry fault.

2. 2 Errors.

Error is the difference between the expected output and the actual output of a system. A system is said to perform erroneously when it starts behaving in a manner that is against its specification and compliance. To study the nature of errors in a cloud computing scenario, a few of them have been listed below.

Network Errors.

Cloud is a network of remote servers. Hence, we may observe a lot of errors in the nodes and the links which connect these servers. This kind of errors are called as Network Errors. Mainly network errors can be in the form of three types. Packet Corruption- As a packet moves from one node to another and traverses across various links, there is a fair amount of chance that it might get corrupted due to the system noise. Packet corruption tweaks the original information and might sometimes go unnoticed.? Packet Loss- If a packet fails to reach its destination, this leads to Packet Loss. The main causes of packet failure are link congestion, device failure, (router/switch) and, faulty cabling. Network Congestion- When the traffic This issue is encountered due to low bandwidth. When the flow of traffic increases on a single path, this may also create network congestion. This issue is very important as it determines the Quality of Service(QoS).

Software Errors.

Software errors are broadly categorized as memory leaks and numerical exceptions. Memory Leaks- When there is a bug in the software wherein the application uses huge memory to perform the task but the memory, which is no longer needed, is not freed upon the completion of the task. Numerical

Exception-A software does a lot of numerical computations which are required by the applications. The applications might sometimes generate issues due to some numerical conversions which raise exceptions. If these exceptions remain unhandled then errors persist in the system.

Time Based Errors.

These errors arise when applications do not complete their task execution in a time bounded manner. This can be subdivided into three types. Transient Errors- the probability of occurrence is very less. Intermittent Errors- The pattern of these errors is sporadic but observed many number of times. Permanent Errors- These occur more number of times with a deterministic pattern.

2. 3 Failures.

As said earlier, failures result due to errors. If a system does not achieve its intended objective, then it’s in a state of failure. Several things can go wrong in a system and yet the system may produce desired results. Until the system produces wrong output, there is no failure? 4 To study the nature of failures, following is the list of failures.

Node Failure.

In distributed systems, such as cloud computing, we see that sometimes resources and nodes are dynamically added to the system. This brings along a lot of uncertainties and the chances of node failure increase. Reliability and availability are the major criteria for nodes to be adjudged as functioning properly. Node failure occurs if a node is not available at any time a node is

not present in the system to perform tasks(unavailable) or produces errors while doing computations.

Process Failure.

Process failure occurs when a process is unable to place the messages into the communication channel and transmit it or a process's algorithm is unable to retrieve messages from the communication channel?

Network Failure.

Network failures are very serious issues with regards to cloud computing. There is no communication without a network. Network failures occur when there is a link failure, network device failures such as routers and switches, configuration changes in a network. Configuration change or a change in policy of a machine will cause problems to the applications using the resources of that machine and this problem is most likely reason for a network failure.?

Host Failure.

A host is a computer that communicates with other computers on the network. In the scope of Cloud Computing, hosts are servers/clients that send/receive data. Whenever a host fails to send the requested data due to crashes, host failure occurs.????

Application Failure.

Cloud applications are the software codes that run on cloud. Whenever bugs develop in the codes, application fails to fulfil its intended objective. The errors caused due to this leads to Application Failure. # cloud endure……

3. Reactive Fault Tolerance Strategies.

Fault Tolerance Strategies in Cloud Computing are of two types, namely, Proactive, and Reactive. Proactive Fault Tolerance Strategies are those techniques which help in anticipating faults and provides preventive measures to avoid the occurrence of faults. Here, the faulty components in the system are identified and replaced with operational ones. Reactive Fault Tolerance Strategies, are the techniques used to effectively troubleshoot a system upon occurrence of failure(s). Various reactive fault tolerance strategies are discussed below.

3. 1 Checkpointing.

In Checkpointing, the system state is saved and stored in the form of checkpoints. This taking is both preventive and reactive. Whenever a system fails, it rolls back to the most recent checkpoint. This is a popular fault tolerance technique and placing the checkpoints at appropriate intervals is very important.

Full Checkpointing.

Complete state of the application in saved and stored at regular intervals. The drawback of full checkpointing is that it needs a lot of time to save and requires huge chunk of storage-space to save the state.

Incremental Checkpointing.

This is an improvement over the full checkpointing. This method performs full checkpointing initially and thereafter only the modified pages of information from the previous checkpoint are stored. This is much faster and reliable than full checkpointing.

Optimized Checkpoint/Restart.

The crux of checkpointing lies in how we space our checkpoints. Good number of checkpoints ensure that the application is resilient to failure. However, this comes at the cost of time, space, and causes a lot of overhead. On the other hand, having less number of checkpoints makes our application vulnerable to faults thereby causing failure. It has been seen that cloud tasks are typically smaller than the grid jobs and hence more time sensitive to the checkpointing/restart cost.? Also, characterizing the failures in the cloud tasks using a failure probability distribution function will be inaccurate as the task lengths in cloud tasks depend on the user priority too.? This technique aims at bettering the performance of Checkpointing technique in threefold approach. Firstly, optimize the number of checkpoints for each task. Secondly, as the priority of the task may change during its execution, a dynamic mechanism must be designed to tune the optimal solution in the first step. Thirdly, find a proper tradeoff between local disks and shared disks to store the checkpoints. The optimal number of checkpoints is calculated by evenly spreading the checkpoints during the execution of the task. The calculation is done without modelling the failures using a failure probability distribution function. A key observation that we

make during the execution of cloud tasks is the tasks with higher priority have longer uninterrupted execution lengths in comparison with low-priority tasks. Hence the solution needs to be more adaptive considering the priority of the tasks. Mere equal spacing of the checkpoints will not do in this case. If the priority of the task remains unchanged the Mean Number of Failures(MNOF) remains the same. The position of the next checkpoint needs to be recalculated and its position needs to be changed if the priority factor that influences the MNOF changes during the execution of the task. Lastly, the problem of where to store the checkpoints is addressed. The checkpointing costs for both local disks and shared disk is calculated and then based upon the costs an efficient choice is made. It is noticed that, as the memory size of the tasks increase, the checkpointing costs also increase. Also, when multiple checkpointing is done, in the local disks, there is no significant increase in the costs, but owing to congestion, there is a significant rise in checkpointing costs. Hence, a distributively-managed algorithm is designed to mitigate the bottleneck problem and lower the checkpointing costs.

3. 2 Retry.

Simplest of all the fault tolerance techniques. The task is restarted on the same resource upon occurrence of the problem. The underlying assumption behind this approach is that during the subsequent attempts, the problem will not show up.?

3. 3 Task Resubmission.

A job consists of several small tasks. When one of the tasks is failed, the entire job gets affected. In this technique, the failed task is resubmitted either to the same resource or a different one to finish the execution of the task.

3. 4 Replication.

Running the same task on several machines which are different locations. This is done to ensure that when a machine fails, the process of task execution is not halted as the other machine takes it up. Replication is further categorized as follows.

Semi-active Replication.

The input is provided to all the replica machines. The task execution simultaneously goes on in the primary replica as well as the backup replica. However, the primary replica only provides the output. When the primary replica goes down, the backup replica provides output. This technique uses a lot of network resources as the task is running in simultaneously in all the replicas. VMware uses Semi-active replication Fault Tolerance Strategy. [4.]

Semi-passive Replication.

This technique has a flavor of checkpointing in addition to replication. The main replica performs the checkpointing operation over the state information. Replication is done by transferring this checkpoint information to all the backup replicas. The backup machines don’t have to concurrently execute the task with the primary replica, but its duty is to save the latest checkpoint information. When the primary replica fails, it designates the

backup replica to takeover. The checkpoint information is updated with some loss in the execution. This technique uses lesser network resources than the semi-active replication but there is a tradeoff as some of the execution. Also, in this case, whenever the backup fails, the latency is more as the time taken for recovery and reconfiguration when compared with semi-active replication. [ref 3]

Passive Replication.

The state information is stored in the form of checkpoints in a dedicated backup machine. When the backup fails, the Fault Tolerance Manager, commissions another machine to be the backup. The backup is updated by restoring the last saved checkpoint. The fault tolerance manager uses a priority based scheme while appointing new backups.

3. 5 Job Migration.

When a task fails in one of the machine, it can be transferred to another virtual machine. Sometimes, if a task in a job cannot be executed due computational and memory constraints, the task is given to another machine to execute.

3. 6 Rescue Workflow.

A cloud job consists of several small tasks. Upon failure of a task, this method continues the execution of the other tasks. The overall workflow is stopped only when the failure of the task impacts the entire job. [rescue workflow]

4. Comparative Summary of the Reactive Fault Tolerance Strategies.

Checkpointing: This technique effectively detects Application Failure. This technique is used when the application size or the task size is too big. Moreover, checkpointing provides efficient resource utilization.

Retry: If the problem persists beyond multiple tries, this method is time inefficient. This is used to detect Host failure and Network failure.

Task Resubmission: As the job is tried on the same or different resource, this technique is both time consuming and has more resource utilization. This detects Node Failure and Application Failure.

Replication: This technique detects Node Failure and Process Failure. As the task is run on various machines, we see more resource utilization here.

Job Migration: This technique detects Node and Process failures. This method is time efficient as the task which cannot be executed in a machine is transferred to another.

Rescue-Workflow: This method detects Node failure and Application failure. This is a time-inefficient technique.