Fault Tolerant Server Using Map Reduce Technique

Sujit Prahlad prasad

Student, Department of MCA, ASM IMCOST, Maharashtra, India

---***---

Abstract - *In this paper, we explore the unique strategies of fault tolerance which are utilized in server groups. This paper focused on the techniques used to detect and recover from different types of a fault occurring in the servers. A deficiency can happen because of a connection failure, resource catastrophe, or by some other motive to endure the working of servers easily and precisely. These flaws can be recognized and improved through numerous strategies. A proper fault indicator can avoid the loss because of a server crash and consistent fault tolerance technique can spare from a server failure. This paper gives how these strategies are applied to identify and endure faults from Server Groups.*

Key Words: Fault Tolerance, Fault Detection, Recovery techniques

1. INTRODUCTION

Servers that are subject to the failure of elements in the system are known to be Fault tolerance which may also be called as fail-safe design. A fault-tolerant server may keep on working to operate just fine even after errors occur such as power supplies fail or absence of resources. Fault in the system due to any reason can cause serious damages to ongoing tasks. Tasks running on the server should be scalable, reliable, and feasible. Server systems such as robotics, air traffic control systems, nuclear power plant system, etc. are exceptionally responsive to be on time. If any errors not detected and recovered on time, it can fail the system in the server groups. Even under hardware and software faults, these systems must have the capacity to function with high availability. Adaptation to non-critical failure is the significant procedure used to keep up reliability in these systems.

Well-known effective methods for fault tolerance are Hardware and software redundancy. Hardware fault-tolerance accomplished through applying additional equipment like processors, Ad-hoc network, memory management where as in software fault tolerance to non-critical failure detector, messages are added into the system to manage deficiencies. Flaw ought to be identified by applying a consistent fault detector followed by some recovery strategy. Many fault detection strategies are accessible yet it is important to apply a suitable fault detector. Inconsistent fault detector can commit error by incorrectly presuming the right procedure or trusting the damaged procedure [1].

2. CHARACTERISTICS FAULT TOLERANT SERVERS

A fault-tolerant server may have one or more of the following characteristics:

2.1 Fault containment:

At the point when a failure happens, it might bring about harm to different components inside the system, in this way making a second or third flaw and causes system failure. For instance, if a simple circuit flops it might build the current over the system harming rationale circuits unfit to withstand high current conditions. Fault regulation is to evade or limit collateral damage brought by a single point failure.

2.2 No Single Point Repair Takes the System Down:

Expanding the single point failure doesn't require shutting down the system due to a failed component, for instance. It additionally implies the system stays on the web and operational during the fix. This may present difficulties for both the structure and the maintenance of a system. Hot-swappable force supplies are a case of a fixed activity that keeps the system working while at the same time displacing a defective power supply.

2.3 No Single Point of Failure:

This implies if a capacitor, block of programming code, an engine, or any single thing comes up short, at that point the system doesn't fall flat. For instance, numerous emergency clinics have a reinforcement power system on the off chance that the grid power fails, in this way keeping system inside the medical clinic operational. These systems may have different repetitive plans to keep up an elevated level of adaptation to non-critical failure and flexibility.

2.4 Fault isolation or identification:

The system can distinguish when a fault happens inside the framework and doesn't allow the broken component to badly impact to useful capability (for example Losing information or making rationale mistakes in a financial system). The defective components are distinguished and separated. Certain portions of the system may have the sole motivation behind identifying faults.

2.5 Robustness or Variability Control:

At the point when a system encounters a single point failure, the framework changes. The change may cause transient or perpetual changes influencing how the functioning components of the framework reaction and capacity. Dissimilarity occurs, and when a failure happens there frequently is an expansion in fluctuation [2]. For instance, when one of two force supplies come up short, the rest of the force gracefully assumes the full heap of the force request. This change ought to happen without affecting the performance of the system. The capacity to structure and assembling a robust system may include a plan for six sigma, a plan of trial streamlining, and different devices to make a system ready to work when a failure happens.

In some cases, the system may be able to operators with no or only minimal loss of functional capability, or the reversion operation significantly restricts the system operation to a critical few functions.

Causes:

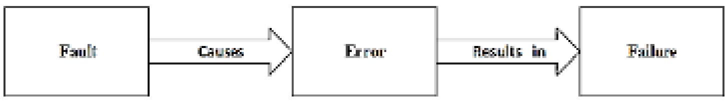

Fig. 1. Causes of Failure [2]

As we can see from Figure 1, the server failure was caused by a system fault. Fault tolerance refers to the techniques and behaviors that are followed to prevent or counteract software defects created in the system after development, or to handling errors during compilation. Mechanisms such as handling software systems and dealing with faults whenever it occurs to prevent system failure and ensure continuity of services are techniques provided by these fault management. Through the implementation of fault tolerance in the cloud, the major benefits we take are recovery from failures, enhanced performance metrics, and cost reduction. Better conceptual understanding of fault-tolerant algorithms and techniques used in Fault-tolerant Server Groups are described in this paper. This paper also has its proposed model for fault tolerance in cloud computing [2].

3. TYPES OF FAULT

Various types of faults can occur in Server Groups. These defects can be classified on several factors such as:

Network fault: Network fault occurs due to network segmentation, destination failure, link failure, packet corruption, packet loss, etc.

Physical faults: This error can occur in hardware such as a CPU, a memory error, a storage error. Media bugs: Media head malfunctions are a mistake. Processor Errors: A processor fault is caused by operating system crashes. Process Errors: Error caused by resource shortages, software errors.

Service expiry fault: The service time of the resource may expire while the application is using it. The fault can be classified based on computing resources and time. Failures occurring during computation on system resources can be classified as: time failure, missed failure, response failure, crash failure, and response failure.

Permanent: These failures are caused by accidentally cutting a wire, electrical breakdown, and so on. The copy of these failures is easy to be repeated. Major disruption can be caused by these failures and some parts of the system may not function as desired.

Intermittent: These failures appear occasionally. Most of these failures are ignored when checking the system and only appear when the system is powered on. Therefore, it is difficult to estimate how much damage these failures can cause to the system.

Transient: These failures are caused by some faults inherent in the system. However, these failures can be fixed by restarting the system to a previous state, such as restarting the software or sending the message back. In Computer systems these are very common failures.

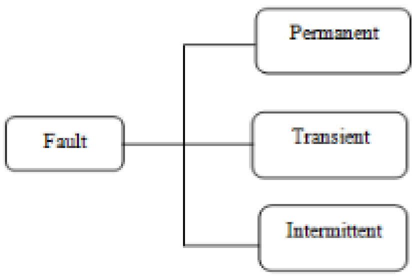

Fig. 2. Types of Fault in a system [3]

The hardware fault tolerance is the main focus in a Server Groups, often software fault tolerance is ignored [3]. This is indeed surprising because the hardware components have a much higher reliability than the software running on them. Most system designers go to great lengths to limit the impact of a hardware failure on system performance.

There are several approaches to fault tolerance in Server Groups. A mistake can be tolerated depending on the manner of its behavior or event.

The following are the methods of fault tolerance in a Server Group.

3.1 Replication
  3.1.1 Job Replication
  3.1.2 Component Replication
  3.1.3 Data Replication
3.2 Check-pointing
3.3 Scheduling/ Redundancy

3.1 Replication:

Replication is the system of sharing records by which it can make sure consistency between redundant resources (i.e. software or hardware components) to enhance reliability (fault-tolerance or access). Job replication is a way of replicating jobs on a couple of servers which includes a network able to receiving jobs in a grid computing service, executing them, performing checksum operations on them, and sending them again to the purchaser displaying a distributed community in which each server can do S is. Contact every other and each server is attached to a couple of purchasers C. A job may be delivered to the server for the operation and the result is returned to the consumer. Data substitute is usually used by fault tolerance mechanisms to growth availability in grids consisting of environments where failures are extra frequent

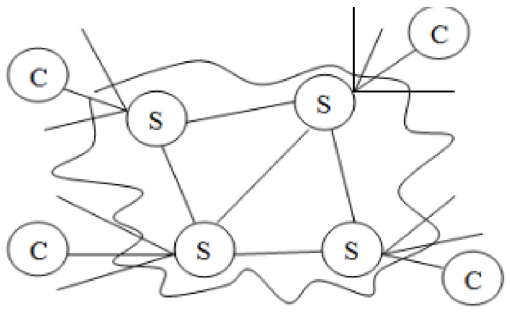

Fig. 3. Server Groups

This record is saved on more than one storage gadget as a replica. Data replication may depend on synchronous or asynchronous statistics compatibility [4]. The additives are replicated on special machines, and if a component or system fails, that application may be moved and run on another gadget with the required components.

3.2 Check-pointing:

It is the process of saving from the complete execution of a task. It checks the acceptance test, if unsuccessful then go to the previous checkpoint instead of the beginning. A probe point can be a system level, application level, or mixed level depending on its characteristics. Check-pointing is likewise categorized on the premise of in-transit or orphaned messages as proven in parent 3. These are Uncoordinated Checkpointing, Coordinated Check-Pointing, and Communication-Inspired Check-Pointing. Check-pointing can also be classified, which is based on the devices that execute the application and re-establish the application execution state. These are manual code entry, pre-compiler Checkpointing, post-compiler check-pointing [3].

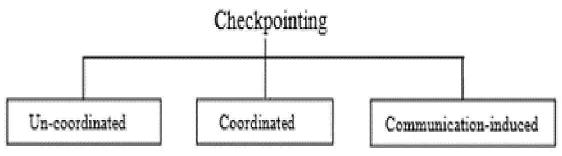

Fig. 4. Types of Checkpointing

The checkpoint can be local or global depending on its size. The check-point of a separate process is the local test point and the active point of setting processes is called the global check-point. Check-pointing has certain deviations such as time-out due to pointing, even if there is no danger. When replacing a test site, the cost of compiling static storage test data points is called the check-pointing cost.

3.3 Scheduling:

This is one of the methods to tolerate fault from the distributed system. It is used to overcome the drawbacks of check-pointing in allotted environments. It is assessed as each time-sharing scheduling, area-sharing scheduling, and hybrid combinations. Scheduling is used for load balancing as well as fault tolerance in server businesses based totally on space or time-sharing. There are three procedures for scheduling such as area, time, and hybrid. Space scheduling is used to tolerate an everlasting or hardware type fault from a system. The primary-backup method is implemented in space redundancy. Time redundancy is used when there is an intermittent type of defect in the system. And hybrid redundancy is used when both are needed. There are some important methods for tolerating defects in various systems given through many authors of their research [4].

4. PROPOSED METHODOLOGY

In case the JobTracker does no longer get hold of any heartbeat from a TaskTracker for a specified duration of time (through default, its miles set to ten minutes), the JobTracker understands that the work related to that TaskTracker has failed. When this situation happens, the JobTracker desires to reschedule all pending and in-progress obligations to some other TaskTracker, due to the fact the intermediate data belonging to the failed TaskTracker might not be to be had anymore.

All finished map obligations need also to be rescheduled if they belong to incomplete jobs, due to the fact the intermediate outcomes residing inside the failed TaskTracker report device won't be on hand to the reduce task.

A TaskTracker can also be blacklisted. In this case, the blacklisted TaskTracker stays in conversation with the JobTracker, but no responsibilities are assigned to the corresponding worker. When a given range of tasks (by default, this range is ready to 4) belonging to a specific process managed by using a TaskTracker fails, the system considers that a fault has occurred [6-7].

Some of the relevant information in the heartbeats the TaskTracker sends to the JobTracker is:

- The TaskTrackerStatus
- Restarted
- If it is the first heartbeat
- If the node requires more tasks to execute

The TaskTrackerStatus contains information approximately the employee managed with the aid of the TaskTracker, such as available digital and bodily reminiscence and facts about the CPU. The JobTracker keeps the blacklist with the defective TaskTracker and also the closing heartbeat acquired from that TaskTracker [7]. So, when a brand new restarted/first heartbeat is obtained, the JobTracker, by using the usage of this statistics, may determine whether or not to restart the TaskTracker or to dispose of the TaskTracker from the blacklist.

After that, the reputation of the TaskTracker is up to date in the JobTracker and a Heartbeat Response is created. This Heartbeat Response includes the following movements to be taken by using the TaskTracker. If there are obligations to perform, the TaskTracker calls for new obligations (that is a parameter of the Heartbeat) and it isn't always in the blacklist, then clean-up duties and setup tasks are created (the clean-up/setup mechanisms have now not been besides investigated yet). In case there aren't clean-up or setup tasks to perform, the JobTracker gets new tasks [7]. When duties are available, the Lunch Task Action is encapsulated in each of them, after which the JobTracker also seems up for:

-Tasks to be killed
-Jobs to kill/clean-up
-Tasks whose output has not yet been saved.

All these moves, if they apply, are introduced to the list of moves to be sent within the Heartbeat Response. The fault tolerance mechanisms applied in Hadoop are confined to reassign responsibilities whilst a given execution fails. In this situation, two situations are supported:

1. In case a venture assigned to a given TaskTracker fails, a conversation thru the Heartbeat is used to inform the JobTracker, so that you can reassign the challenge to another node if possible.
2. If a TaskTracker fails, the JobTracker will notice the defective situation as it will not obtain the Heartbeats from that TaskTracker. Then, the JobTracker will assign the duties the TaskTracker needed to every other TaskTracker. There is also an unmarried point of failure inside the JobTracker, because if it fails, the whole execution fails [6].

The main blessings of the standard approach for fault tolerance applied in Hadoop is composed on its simplicity and that it seems to work well in local clusters, However, the standard method is now not sufficient for large disbursed infrastructures the space between nodes can be too big, and the time lost in reassigning a mission may gradual the machine.

The Master node (Name Node) is an unmarried point of failure in Hadoop. If it goes down, the device is unavailable. Slave (Computational) node failures are fine, and anything walking on them on the time of failure is sincerely rerun on a specific node. In fact, this may arise even if a node is jogging slowly.

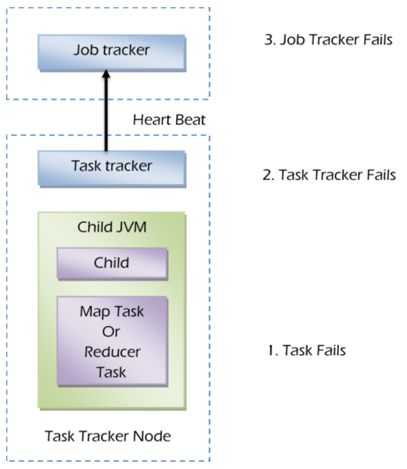

Fig. 5. System Design of places where Failures Occurs [6]

There are mostly three places where the failures can happen. Those are [7]:

- Task Failure
- Task tracker failure
- Job tracker failure

4.1 Task Failure:

If the child task fails, the child JVM reports a project tracker before it exists. The attempt is to be freed up a slot for another project. If the child venture hangs, it is going to be killed. The job tracker reschedules the venture on any other machine. If the non-stop to fails, a job may be failed.

4.2 Task tracker failure:

When a task tracker is working, the task tracker sends the heartbeat to the job tracker. When the job tracker doesn't receive any heartbeat from the task tracker it assumes the task tracker has failed [6]. To schedule tasks on the task, the tracker is removed from the pool of trackers by job trackers. The task tracker is considered down when job tracker does not receive a heartbeat report from any one of them and assign the same task to another idle task tracker by job tracker.

4.3 Job tracker failure:

This is a single point of failure. If the job tracker fails, the entire job will be failed. Chances of getting corrupted/bad records due to any flaw in the logic written in both mapper & reducer and the task will fail because of those records. In the failure case, the job tracker will execute the failed records 4 times by default. In the 4 attempts, if the same job fails all the time then the job tracker will mark the entire job as failed.

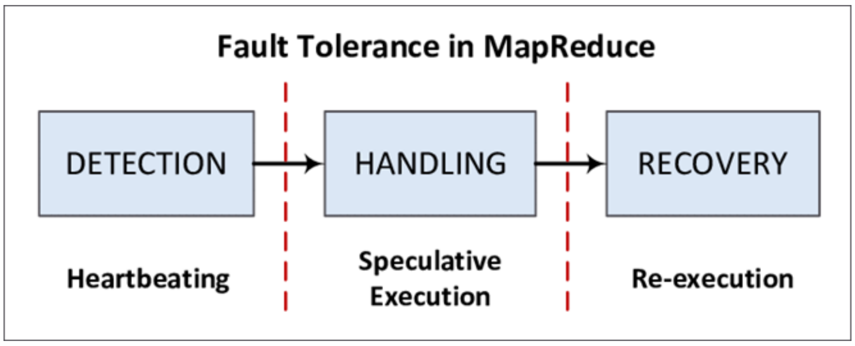

Fig. 6. MapReduce Fault tolerance [1]

5. CONCLUSIONS

To conclude this paper, we can say that the detection of a malfunction in a distributed system is very different as compared to one-sided. The defect tolerance technique also depends on its occurrence. In this paper, we explore various reliable fault detection and fault tolerance methods. There are three things in a real-time distributed system that must be kept in mind when implementing fault tolerance. These are reliable, scalable, and feasible. The feasibility of a distributed system in real-time of a task is very important because a deadline is set for each task and the task must be finished before or after its deadline, even in the system Malfunction can also occur.

REFERENCES

[1] A Study on Fault ToleranceSolution. **Dr. Meena Kumari and Shaima'a Ghamdan.** 21, 2016, International Journal of Engineering Research & Technology (IJERT), Vol. 4.

[2] Fault Tolerance-Challenges, Techniques andImplementation in Cloud Computing. **Singh, Thakur Kapil, RaviTeja, Tarakarama Godavarthi and Pappla, Padmavathi Srinivasa.** 6, s.l. : International Journal of Scientific and Research Publications, 2013, Vol. 3.

[3] Fault Tolerance in MapReduce: A Survey. **Bunjamin Memishi, Shadi Ibrahim, Mar´ıa S. P´erez and Gabriel Antoniu.** 2016, Springer Link.

[4] Fault Tolerance in Real Time Distributed System. **Arvind Kumar, et al.** 2, 2011, International Journal on Computer Science and Engineering (IJCSE), Vol. 3.

[5] Review on Fault Tolerance Techniques in Cloud Computing. **Zeeshan Amin, Nisha Sethi and Harshpreet Singh.** 18, 2015, International Journal of Computer Applications, Vol. 116.

[6] Fault Tolerance: https://www.tutorialscampus.com/

[7] Fault-Tolerance-MapReduce: https://stackoverflow.com/