CASE STUDY NO : 08

Date	/ /20
Page No.	39

Case Study On :- Study of fault toullerance in Distributed system.

Fault Tollerance -

A system is said to fail when it does not meet its specification. In some cases, such as a supermarket's distributed ordering system, a failure may result in some store running out of canned beans. In other cases, such in a distributed air traffic control system, a failure may be catastrophic. As computers and distributed systems become widely used in safety-critical missions, the need to prevent failures becomes correspondingly greater. In this section we will examine some issues concerning system failures and how they can be avoided. Additional introductory material can be found in (Cristian, 1991; and Nelson, 1990). Gantenbein (1992) has compiled a ~~bibtlig~~ bibliography on the subject.

Any system has two major components. Hardware and software. Fault may occur in either of it. So there are separate techniques for fault-tolerance in both hardware and software.

Hardware Fault - Tolerance Techniques -

Making a hardware fault-tolerance is simple as compared to software fault ~~work~~ tolerance techniques make the hardware

work proper and give correct result even some fault occurs in the hardware part of the system. There are basically two techniques used for hardware fault-tolerance.

1) BIST - BIST stands for Build in self Test. system carries out the test of itself after a certain period of time again and again, that is BIST technique for hardware fault-tolerance. when systems detects a fault, it switches out the faulty components and switches in the redundant of it. System basically reconfigure itself in case of fault occurrence.

2) TMR - TMR is triple modular redundancy. Three redundant copies of critical components are generated and all these three copies are run concurrently. Voting of result of all redundant copies are done and majority result is selected. It can tolerate the occurrence of a single fault at a time.

* Software fault-tolerance techniques -

Software fault-tolerance techniques are used to make the software reliable in the condition of fault occurrence and failure. There are three techniques used in software fault-tolerance. First two techniques are common and are basically an adaptation of

hardware fault-tolerance techniques.

1) N-version Programming-

In N-version programming, N-version of software are developed by N individuals or groups of developes. N-version programming is just like TMR in-hardware fault-tolerance technique. In N-version programming, all the redundant copies are run concurrently and result obtained is different from each processing. The idea of n-version programming is basically to get the all error during development only.

2) Recovery Blocks - Recovery blocks technique is is also like the n-version programming but in recovery block technique, redundant copies are generated using different algorithms only. In recovery block, all the redundant copies are not run concurrently and these copies are run one by one.

3) Check-pointing and Rollback Recovery - This technique is different from above two techniques of software fault-tolerance. In this technique, system is tested each time when we perform some computation. This techniques is basically useful when there is processor failure or data corruption.

Date / /20
Page No. 42

Componend Faults -

Computer systems can fail due to a fault in some componend, such as a processor, memory, I/o device, cable or software. A fault is a malfunction, possibly caused by a design error, a manufacturing error, a programming error, physical damage, deteriorbation in the course of time, harsh environmental conditionals, unexpected inputs, operator error, rodents eating parts of it, and many other causes. Not all faults lead to system failures, but some do.

Faults are generally classified as transient, intermittend or permanend. Tranient faults occur once and then disappear. If the operation is repeated, the fault goes away. A ~~bird~~ bird flying through the beam of a microwave transmitter may cause lost bits on some network. If the transmission times out and is retried, it will probably work the second time.

An intermittend fault occues, then vanishes of its own accord, then reappears and so on. A loose contact on a connector will often cause an intermittend fault. Intermittend faults cause a greate deal of aggravation because they are difficult to diagnose. Typically, whenever the fault doctor shows up, the system works perfectly

Date	/ /20
Page No.	43

Conclusion -

To improve the reliability of a system we can add fault-tolerance mechanisms, so as to tolerate faults that cannot be removed at design-time. However, the resulting rise in complexity increases the probability of software ~~fatu~~ faults being introduced.