Fault Tolerance Techniques

Fault - tolerance is defined informally as the ability of a system to deliver the expected service even in the presence of faults.

A common misconception about real- time computing is that fault-tolerance is orthogonal to real- time requirements. It is often assumed that the availability and reliability requirements of a system can be addressed independent of its timing constraints.

A real-time system may fail to function correctly either because of errors in its hardware and/ or software or because of not responding in time to meet the timing requirements that are usually imposed by its "environment."

Hardware fault is some physical defects that can cause a components to malfunction. Software fault is a bug that can cause a program to fail for a given set of inputs.

An error is a manifestation of a fault. The fault latency is the duration between the onset of a fault and its manifestation as an error.

An error latency is the duration between when an error is produced and when it is either recognized as an error or causes the failure of the system.

Error recovery is the process by which the system attempts to recover from the effects of an error.

Recovery from an error is fundamental to fault tolerance.

Two main forms of recovery :

1. Forward error recovery
2. Backward error recovery

Forward error recovery

Forward recovery, attempt to bring system to a new stable state from which it is possible to proceed (applied in situations where the nature if errors are known and a reset can be applied).

Forward error recovery continues from an erroneous state by making selective corrections to the system state.

This include making safe the controlled environment which may be hazardous or damaged because of the failure.

It is system specific and depends on accurate predictions of the location and cause of errors (i.e., damage assessment).

Examples : Redundant pointers in data structures and the use of self-correcting codes such as Hamming Codes.

Advantage forward - error recovery :

1. Less overhead

Disadvantages of forward recovery :

2. In order to work, all potential errors need to be accounted for up-front.
3. Limited use, i.e. only when impact of faults understood.
4. Cannot be used as general mechanism for error recovery.
5. Design specifically for a particular system.

Backward recovery :

Most extensively used in distributed systems and generally safest. It can be incorporated into middleware layers.

Backward recovery is complicated in the case of process, machine or network failure but no guarantee that same fault may occur again.

It can not be applied to irreversible (non-idempotent) operations, e.g. ATM withdrawal.

Advantage backward - error recovery :

1. Simple to implement
2. Can be used as general recovery mechanism
3. Capable of providing recovery from arbitrary damage.

Disadvantage of backward recovery :

1. Check pointing can be very expensive - especially when errors are very rare.

2. Performance penalty
3. No guarantee that fault does not occur again
4. Some components cannot be recovered

Causes of Failuer

There are three causes of failuer,

1. Errors in the specification or design
2. Defecuts in the components
3. Environmental effects

Mistake in the specification and design are very difficult to guard against. Many hardware failuers and all software failuers occurs such mistake.

If the specification is wrong, everything that processeds fro it, design and implementation, likely to be unsatisfactory.

Fault Types

Fault are classified as temporal behaviours and output behaviours.

1. **Temporal behaviours classification**
    - **Fault are of three types :** Permanent, intermittent and transient.
    - **Transient faults :** These occur once and then disappear. For example, a network message transmission times out but works fine when attempted a second time.
    - **Intermittent faults :** These are the most annoying of component faults. This fault is characterized by a fault occurring, then vanishing again, then occurring. An example of this kind of fault is a loose connection.
    - **Permanent faults :** This fault is persistent : It continues to exist until the faulty component is repaired or replaced. Examples of this fault are disk head crashes, software bugs, and burnt-out hardware.
2. **Output behavioure classification**
    - **Malicious faults :** Inconsistent output.
    - **Nonmalicious fault :** Consistent output errors.

- **Fail stop :** Responds to up to a certain maximum numbers of failuers by simply stoping, rather than putting out incorrect outputs. The component simply stops working. For instance, a hard disk which refuses to read or write.
- **Fail safe :** Its failuer mode is biased so that the application process does not suffer catastrophe upon failuers. A component under too much load is likely to fail. A fail safe system, on detecting a large amount of load, processes each request slower to avoid failure.

3. **Independence and correlation**

- Components failures may be independent and correlated.
- **Independent :** A failure is said to be independent it it does not directly or indirectly cause another failure.
- **Correlated :** If the failure is said to be correlated if they are related in some way.