

## Failure Analysis of Fault-Tolerant Computing Systems in Early Design Stages

By Duane Huffman, Senior Application Engineer

Fault tolerance is an essential architectural attribute for achieving high reliability in computing systems dealing with critical data, such as patient information in the medical industry, accounting information in the banking industry, and flight information in the aircraft industry. In these instances, automatic recovery and reconfiguration mechanisms play a crucial role in ensuring these systems are fault tolerant. In many cases, **failover operation** is a key design element employed to make sure critical computing systems remain operational. Failover operation ensures uninterrupted data flow and operability by transferring the operation from a failed component, such as a server or disk drive, to a similar, redundant component. Modeling of fault tolerant systems, which include failover operation, usually cannot be solved using combinatorial techniques such as fault trees and reliability block diagrams due to the complexity of system interactions. Though Markov analysis can be used, in many cases it is difficult to generate a Markov chain that accurately models the system behavior. A methodology that has been successfully used for modeling fault tolerant systems is a decomposition method that uses both combinatorial and Markov analyses.

### Example System

To explain the decomposition technique, we will use an example system for analysis. A block diagram of our example system is shown in Figure 1. The system is a scalable and fault-tolerant PC-based distributed network storage system. The hardware is mostly off-the-shelf hardware. The software is a mix of new and previously developed software, all of which was generated by the same team.

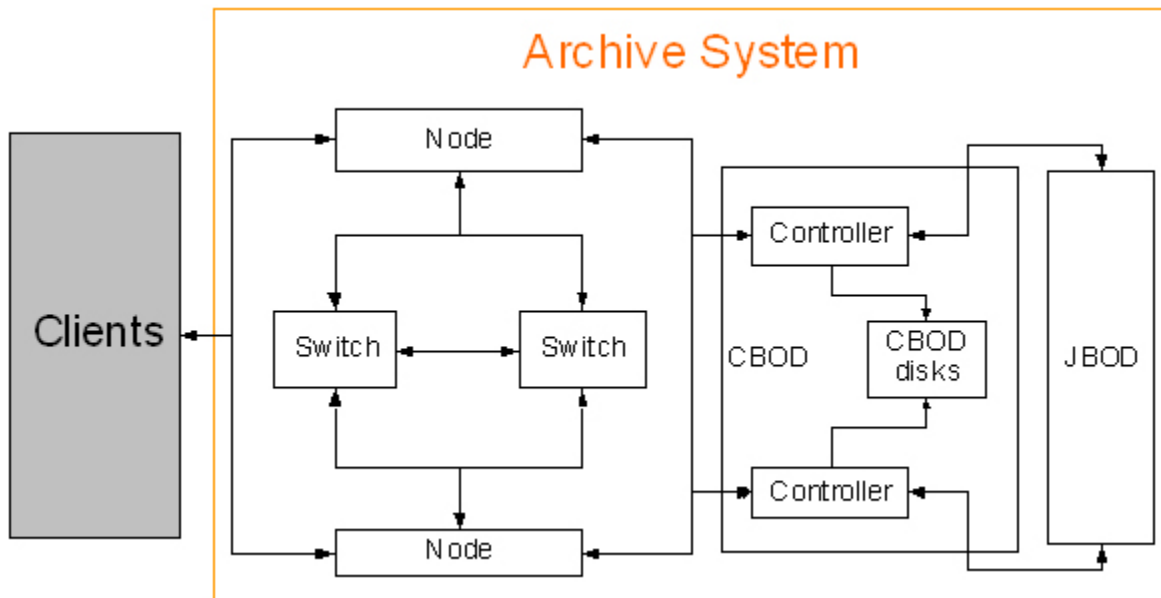


Figure 1. Data Storage System

As with any reliability analysis, the starting point is to understand the key components and subsystems of the system and their required functions. In a fault tolerant system that includes both hardware and software, it is important to understand the contribution of each subsystem to the system reliability. Tools such as functional block diagrams,

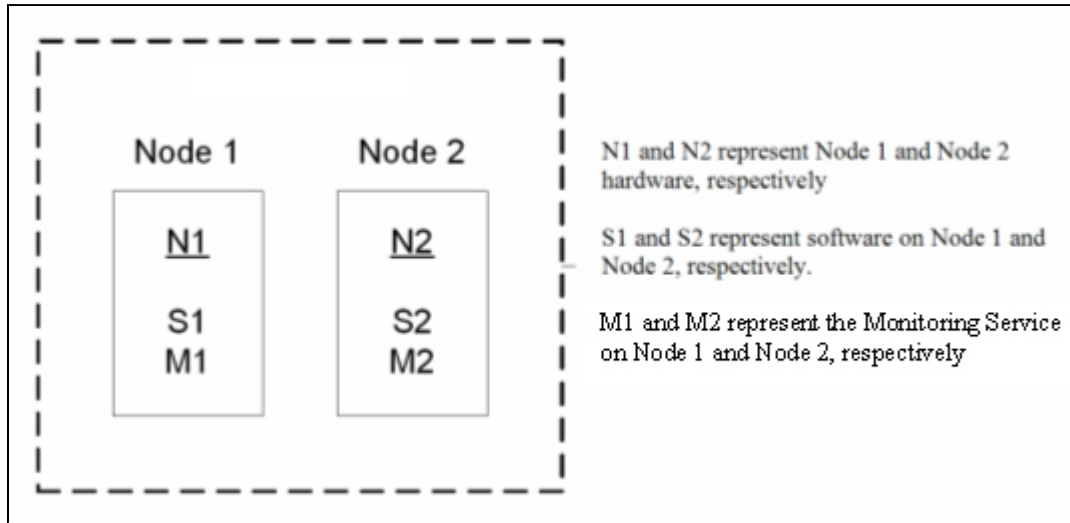
design documents, and flow charts will help to identify the elements and interactions that need to be analyzed. This detailed analysis of the system's function and operation is essential to an accurate analysis.

Consider the Data Storage System shown in Figure 1. For analysis purposes, the system is broken into three main parts: hardware, software, and automatic recovery mechanisms. The hardware part of the system consists of four subsystems, Nodes (servers), Switches (gigabit Ethernet), CBOD (Controllers and CBOD disks), and JBOD (disk storage). The software components of the system, which are not shown in Figure 1, run locally on each node. The software components are the Policy, File Metadata, Object Store, Scheduler, Administration, Security, and Logging subsystems.

The automatic recovery mechanism uses a software monitoring service, which runs on each Node to monitor and maintain system performance. This software is responsible for monitoring the system activity and correcting detrimental situations as they occur. If an uncorrectable event occurs, the service notifies relevant parties. For example, upon the failure of a node, the service will transfer all services from the failed node to another node and then notify designated personnel.

### Gathering R&M Metrics

To begin analysis, each of the three main parts of the system—hardware, software, and automatic recovery mechanisms, is analyzed independently. The failure and repair rates for these subsystems must be determined. For this analysis, the system can be logically represented by the simplified diagram shown in Figure 2.



**Figure 2. Node Subsystem Initial Configuration for Data Storage System**

For the hardware subsystems, failure and repair rate information for parts can be obtained from field data, field returns, and previous analysis performed on similar parts used in these subsystems. In the case of commercially available parts, the manufacturer may provide failure and repair rate information.

For the software subsystems, including the monitoring service, failure information is more challenging to obtain. Several models exist to aid in the analysis of software systems. Many of these models take into account parameters that are not available early in the design stage. For this case, the RAC PRISM RACRates software model can be employed. In the PRISM model, software failure rate is directly proportional to the lines of code (LOC). To estimate the failure rate of the new software system being developed, baseline LOC and failure rate data from similar, previously existing software systems is required. In this example, the PRISM approach is valid because the new software is similar to previously released software and was even developed by the same team. Taking advantage of this PRISM model, with *i* designated as the new product software and BS designated as the baseline software, subsystem failure

rates can be calculated as:

$$\lambda_i = \frac{(LOC)_i}{(LOC)_{BS}} \lambda_{BS}$$

## Analysis

The analysis of the fault-tolerance of the system is based on the function and performance of the monitoring service software, the software subsystems, and the nodes. Because the software subsystems are distributed across the two nodes (Figure 2), there is a complex interrelationship between node and software subsystem failures. The ability of the monitoring service software subsystem to detect failures, either software or hardware, and take the necessary corrective actions must be analyzed. To analyze this subsystem properly, it is necessary to first understand how each hardware and software subsystem can fail. Once all of the failure modes of the system are understood, analysis must be done by considering how the monitoring service will restore each failed system.

The first step is to identify the conditions where the behavior of an individual subsystem is independent of other subsystems. Once these conditions are identified, each subsystem is analyzed independently. For the example, the independent subsystems to analyze are the node subsystem, the software subsystems, and the monitoring service subsystems. The results of these independent analyses can then be integrated for the overall system analysis. For the failover analysis specifically, the node subsystem is analyzed independently. Then, for each state of the node, the software and monitoring service are analyzed.

The advantages of taking this approach are:

- It requires only three simple Markov models (Figures 3, 4, and 5).
- It is easy to verify and validate.
- It is easy to extend to multiple nodes, software components, and different repair policies.

## Node Failures

The starting point of analysis is to logically decompose the system into two parts, the nodes and the rest of the system. First, only the node subsystems are considered. Then, all other functions of the system are considered based on the states of the node subsystems. The overall system availability can then be calculated using combinatorial techniques that utilize the results of the decomposed Markov chains.

The Markov chain for analysis of the node subsystems is shown in Figure 3. The state probabilities for this chain can be solved using frequency balance equations.

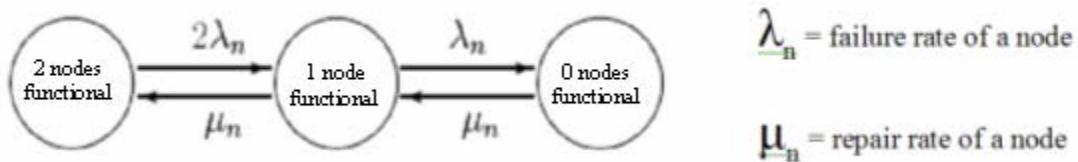


Figure 3. Markov Chain for Node Failure Analysis

Next, consider the function of the monitoring service and the software subsystems for the "2 nodes functional" state of

Figure 3. The Markov chain for this state is shown in Figure 4. When the monitoring services are functioning, it is assumed that a software subsystem failure will be detected and repaired. Further, in these states, the overall system is functional if and only if the software subsystem is functional. Therefore, it is only necessary to consider software subsystem failures explicitly in the two states where the monitoring systems have both failed. The state probabilities can be solved, and the unavailability contribution  $U_0$  can then be calculated.

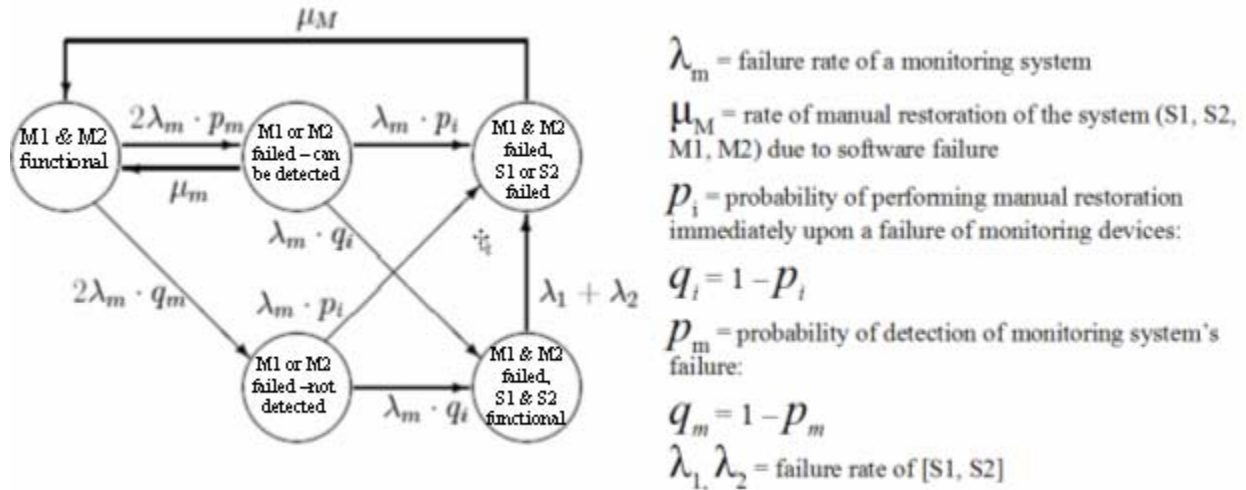


Figure 4. Markov Chain for Software and Monitoring Service Failure Considering Both Nodes Working (State "2 nodes functional" of Figure 3)

Next, consider the function of the monitoring service and the software subsystems for the "1 node functional" state of Figure 3. The Markov chain for this analysis is shown in Figure 5. As in the analysis for the Markov chain shown in Figure 4, it is only necessary to consider software subsystem failures explicitly in the two states where the monitoring systems have both failed. The state probabilities can be solved, and the unavailability contribution  $U_1$  can then be calculated.

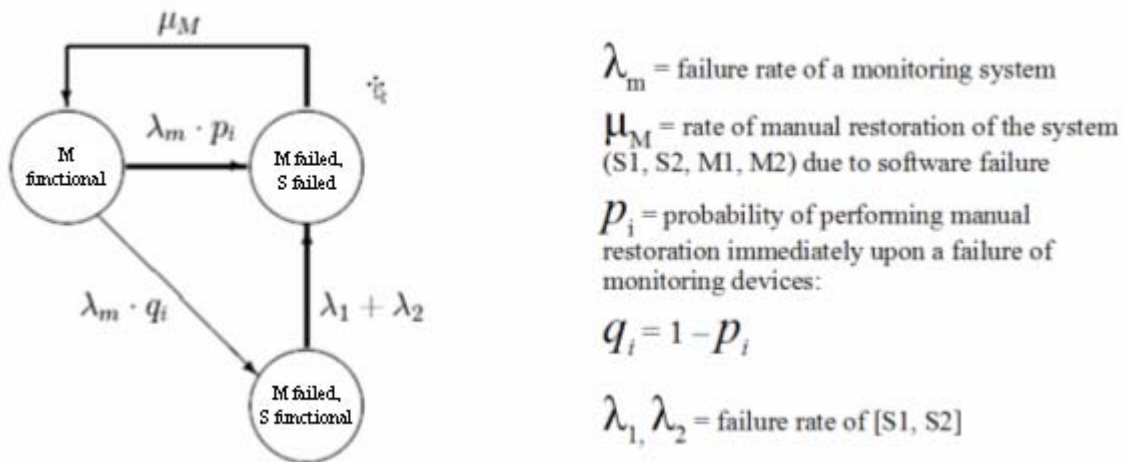


Figure 5. Markov Chain for Software and Monitoring Service Failure Considering One Node Working (State "1 node functional" of Figure 3)

Lastly, consider the "0 nodes functional" state of Figure 3. When both nodes have failed, the system is in a failed state. Therefore, it is not necessary to analyze the subsystems. The state probability can be solved and is a direct contribution to system unavailability ( $U_2$ ) because this state is a system failure state.

## Computing System Performance Metrics

The unavailability of the system is calculated as the sum of unavailabilities of the system at different mutually exclusive states.

$$U = U_{\text{unavailability during failover operation}} + U_{\text{unavailability during normal operation}}$$

The unavailability of the system is calculated as the sum of unavailabilities of the system at different mutually exclusive states.

$$U = \sum_{i=0}^2 \left\{ \text{System unavailability when exactly } i \text{ nodes are functioning} \right\}$$

The unavailability during failover operation (when one out of two nodes is failed) is the sum of the unavailability contributions of the following events:

1. Failover of N1 is successful. This event occurs when N1 is failed and both S1 and M2 are good. It is an automatic recovery process. The unavailability associated with this process is  $U_3$
2. Failover of N2 is successful. The unavailability associated with this process is  $U_4$
3. Failover of N1 is failed. This event occurs when N1 is failed and either S1 or M2 are failed. It is a manual restoration process. The unavailability associated with this process is  $U_5$
4. Failover of N2 is failed. The unavailability associated with this process is  $U_6$

## Unavailability Calculations

Overall system unavailability is ( $U$ ) can be computed as:

$$U = U_0 + U_1 + U_2 + U_3 + U_4 + U_5 + U_6$$

If you consider the unavailability as a result of failures that require manual restoration, then  $U_3$  and  $U_4$  do not need to be considered. Therefore, the unavailability due to critical failures is:

$$U = U_0 + U_1 + U_2 + U_5 + U_6$$

---

---

---

© Copyright 2008 Relex Software Corporation