the Availability Digest

Calculating Availability – The Three Rs

December 2006

Introduction

In our previous two articles on calculating availability, we focused on redundant systems that could suffer two or more subsystem failures depending upon how many subsystem spares were provided. We assumed that if the system went down because it had lost all of its spare subsystems plus one more, it could be returned to service as soon as the first failed subsystem was repaired.

However, things are more complicated than this in active/active systems. Not only must a node (a subsystem) be *repaired*, but it must be *recovered*, and the system *restored* to an operational state before services are once again being provided to the users.

In our first article on Calculating Availability – Redundant Systems, we derived the basic availability equation for an active/active system:

$$F = \frac{MTR}{MTBF} = f(1-a)^{s+1} \tag{1}$$

where

F	is the probability of failure for the system.
MTR	is the mean time to repair the system (its average repair time).
MTBF	is the mean time before failure of the system (its average uptime).
a	is the availability of a node (the proportion of the time that it is up).
s	is the number of spares in the system (it takes the failure of *s*+1 nodes to take down the system).
f	is the number of ways that *s*+1 nodes can fail (the number of failure modes).

In our previous article on Calculating Availability – Repair Strategies, we explored the differences between parallel and sequential repair strategies on system availability. When compared to sequential repair, we showed that parallel repair reduces the probability of failure for a singly-spared system by a factor of two and for a dually-spared system by a factor of six [basically (*s*+1)!], where *s* is the number of spares].

These relations were based on the assumption that following a system failure, *as soon as any one of the failed nodes was repaired, it was returned to service; and the system was up and running*.

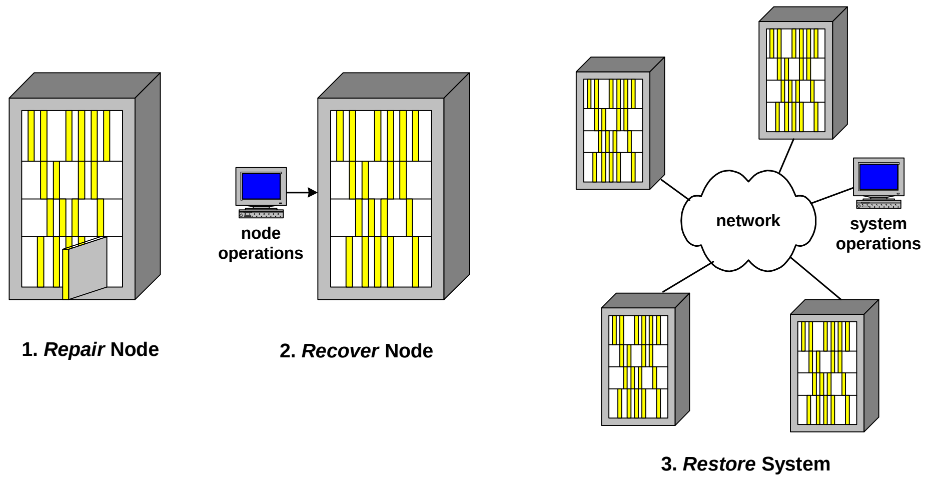
The Three Rs

However, getting the system up and running is more than just a matter of repairing one of the failed nodes. In fact, there are three “r”s to consider relative to returning the system to service:[1]

- **repair –** The fault that caused the node to fail must be *repaired*. This usually entails a hardware replacement.

- **recovery –** Once the node is repaired, it must be *recovered*. This might require that the software environment be reloaded, the applications started, and the databases be opened, among other tasks. Upon the completion of node recovery, system restoration can begin.

- **restore –** Once one of the failed nodes is fully recovered, the system has the node complement that it needs to be put back into service.. In some cases, service can be *restored* to the users at this time. However, this may not always be true. For example, the failed node’s database will usually have to be resynchronized with the surviving databases. In some cases, a backlog of manually-completed transactions may have to be entered before system operation can be considered to be restored to its normal state.



Hardware Repair

In our previous article, we considered two nodal repair strategies, sequential repair and parallel repair:

Under the *parallel repair* strategy, service technicians are dispatched to all failed nodes and repair them simultaneously. The first node that is repaired returns the system to service.

Under the *sequential repair* strategy, there is only one service technician, who repairs the nodes one at a time. When he has completed the repair of the first node, the system can be returned to service.

This led to the following rules for repair strategy:

[1] The impact of repair, recovery, and restore on availability is discussed in more detail in the forthcoming book, *Breaking the Availability Barrier: Achieving Century Uptimes with Active/Active Systems*, by Paul J. Holenstein, Dr. Bill Highleyman, and Dr. Bruce Holenstein.

Parallel Repair Failure Probability Advantage Rule: If there are *s* spares configured for the system, parallel repair will reduce system failover probability by a factor of (*s*+1)! as compared to sequential repair.

Parallel Repair MTR Advantage Rule: If there are *s* spares configured for the system, parallel repair will reduce system MTR by a factor of (*s*+1) as compared to sequential repair.

Parallel Repair MTBF Advantage Rule: If there are *s* spares configured for the system, parallel repair will increase system MTBF by a factor of *s*! as compared to sequential repair.

We derived availability relations for these cases. The relationships for the singly-spared systems are

$$F = \frac{n(n-1)}{2}(1-a)^2 \quad \text{for parallel repair and one spare node} \qquad (2)$$

$$F = n(n-1)(1-a)^2 \quad \text{for sequential repair and one spare node} \qquad (3)$$

where

F	is the probability of failure for the active/active system.
a	is the availability of a node.
n	is the number of nodes in the system.

Node Recovery

In the above analysis, we focused extensively on one of the three “r”s – *repair*. We assumed that once a node was repaired, it was ready to be put back into service.

If things were only that simple. For once the node is *repaired*, it must next be *recovered* before it can be returned to service and the system returned to operation. Even then, there may be more work required before service can be *restored* to the users.

Recovery of a failed node generally requires many steps and can take hours. Though perhaps not an exhaustive list, these steps may include:

- The application environments (such as transaction monitors) must be brought up.
- The applications must be restarted.
- The applications must open their databases.
- The node must be reconnected to the network.
- The proper operation of the repaired node must be verified.

Only after all of these steps (and perhaps others) have been completed can the node be put into service and the system restore tasks begun.

The node’s mean time to repair, mtr, is the amount of time that the node is down and is the sum of the node’s repair time plus its recovery time. Thus, nodal recovery time adds directly to the nodal hardware repair time to arrive at the node’s mean time to repair. That is, the mean time to repair a node, its mtr, is the sum of its hardware repair time and its recovery time. These are the tasks that the repair person must accomplish before he has completed his repair of the node:

node repair time (mtr) = hardware repair time plus node recovery time

System Restore

Once one of the failed nodes has been recovered, the system is in a position to be brought back online. However, before it can be put into full use, there are still some tasks that may have to be accomplished. For instance, the database of the failed node will probably have to be resynchronized with the surviving database copies. Furthermore, there may have been a backlog of transactions, manually entered during the system outage, which must be reentered before the system can be put into normal operation. Finally, users may have to be rebalanced across the network.

These we define as system *restore* tasks. They differ from node recovery tasks in that there are no parallel efforts being made to accomplish these tasks, unlike node recovery tasks which are being undertaken by repair personnel at each of the failed nodes if parallel repair is being used.

To simplify mathematical notation, let us replace mtr with *r*:

r = mtr = nodal repair time = hardware repair time + node recovery time

We know from our analysis in our previous article that under parallel repair, the average time to get the first node repaired is *r* /(*s*+1), where *s* is the number of spare nodes. The system MTR is the sum of the average nodal repair time plus the time to accomplish the system restoration tasks. Let us call the time required to accomplish the system restore tasks *R*. Then

$$\mathrm{MTR} = r/(s+1) + R \quad \text{for parallel repair} \tag{4}$$

where

MTR	is the system mean time to repair. It is the time from the last node failure to the time that services are restored to the users.
r	is the nodal repair time (hardware repair time plus node recovery time).
s	is the number of spare nodes.
R	is the system restoration time.

In the introduction above, we repeated the availability equation derived in our previous article for repair only:

$$F = \frac{\mathrm{MTR}}{\mathrm{MTBF}} = f(1-a)^{s+1} \tag{1}$$

In our previous analysis for parallel repair, MTR = *r* /(*s*+1). However, we must now adjust this equation to reflect the more accurate value of MTR as given by Equation (4). Since the system failure probability is proportional to MTR, the previously calculated failure probability expressed by Equation (1) must be increased by the factor

$$\frac{r/(s+1)+R}{r/(s+1)}$$

and therefore,[2]

[2] These relationships are formally derived in Appendix 3, Failover Fault Models, *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, by Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein.

$$F = \frac{r/(s+1)+R}{r/(s+1)} f(1-a)^{s+1} \quad \text{for parallel repair} \tag{5}$$

For a singly-spared system (*s* = 1) with parallel repair, this becomes

$$F = \frac{r/2+R}{r/2}\frac{n(n-1)}{2}(1-a)^2 \quad \text{for parallel repair and one spare node} \tag{6}$$

For sequential repair, MTR = *r* + *R*. Following the above argument, we obtain

$$F = \frac{r+R}{r} f(1-a)^{s+1} \quad \text{for sequential repair} \tag{7}$$

For a singly-spared system with sequential repair, this becomes

$$F = \frac{r+R}{r} n(n-1)(1-a)^2 \quad \text{for sequential repair and one spare node} \tag{8}$$

In Equations (5) through (8),

F	is the probability of failure of the active/active system.
a	is the availability of a node.
r	is the repair time for a node (hardware repair plus recovery).
R	is the system restoration time.
n	is the number of nodes in the system.
s	is the number of spare nodes provided for the system.
f	is the number of ways that *s*+1 out of *n* nodes can fail

Which Are Recovery Tasks and Which Are Restore Tasks?

There may be some confusion as to whether a task should be included in node recovery time *r* or in system restore time *R*. The answer to this quandary is given by Equation (5).

For parallel repair, if a task is being done by a repair technician at one node site, similar tasks are being done by repair technicians at the other repair sites. In this case, the task becomes part of nodal repair and enjoys the 1/(*s*+1) economy of parallel repair. Therefore, it is a node recovery task and is included in the node repair time *r* as a recovery task.

However, if it is a task that is done centrally once the first node is recovered, it does not benefit by parallel repair and is included in the system restore time *R*.

For instance, loading the applications into a node is definitely a task that is accomplished by the repair technician at the repair site. Therefore, it contributes to node recovery time.

Reentering backlogged transactions is a nonparallel activity and is a system restore task.

Synchronizing the failed node's database may either by done by the repair technician on-site or may be done by central operations personnel. In the former case, it is a node recovery task and is included in *r*. In the latter case, it is a system restore task and is included in *R*.

Note that if sequential repair is used, there is no parallel advantage. As seen by Equation (7), there is no difference between a nodal recovery task and a system restore task so far as the system failure probability *F* is concerned. This is because all tasks are being done one at a time anyway.

Summary

We have explained why simply *repairing* one of the downed nodes is insufficient to return the system to service. The node must also be *recovered*, and some actions requiring use of the fully functional system may be needed before service to the users can be *restored*. The availability relations for these cases are given in Equations (5) through (8).