



**“Achieving Century Uptimes”  
An Informational Series on Enterprise  
Computing**

**As Seen in *The Connection*, An ITUG Publication  
December 2006 – Present**

**About the Authors:**

Dr. Bill Highleyman, Paul J. Holenstein, and Dr. Bruce Holenstein, have a combined experience of over 90 years in the implementation of fault-tolerant, highly available computing systems. This experience ranges from the early days of custom redundant systems to today’s fault-tolerant offerings from HP (NonStop) and Stratus.

***Gravic, Inc.***  
Shadowbase Products Group  
301 Lindenwood Drive, Suite 100  
Malvern, PA 19355  
610-647-6250  
<http://www.gravic.com/shadowbase>

# Achieving Century Uptimes

## Part 11: The Rules of Availability II

July/August 2008

Dr. Bill Highleyman  
Dr. Bruce Holenstein  
Paul J. Holenstein

We continue in this article a review of our “Rules of Availability,” as published in our series of books entitled *Breaking the Availability Barrier*.<sup>1</sup> We choose those rules that are particularly applicable as *best practices* to achieve continuous availability with redundant systems, with a focus on active/active systems.

### The Facts of Life

**Rule 18:** *Redundant hardware systems have an availability of five to six nines. Software and people reduce this to four nines or less.*

Even Microsoft touts five nines of availability in their ads. But they are referring to availability at the system level, not at an operational level. The extreme in hardware availability is achieved by NonStop Integrity TMR (triple modular redundancy) servers that theoretically can achieve seven nines of hardware availability.

Hardware availability has ceased to be a significant part of the system availability equation. It is application bugs and people errors that now control availability. This means that thorough application testing, documented operational procedures, and good operator training represent the path to improving your system availability.

**Rule 19:** *(Bartlett’s Law) - When things go wrong, people get stupider.*

This is probably one of the most important availability rules of all, and this rule plays a major factor in Rule 18 above. When a system crashes, the pressure on the operations staff can rise to untenable levels. Not only are the operations people trying to determine what caused the problem, but management is suddenly very much involved and is demanding answers and a course of action. What is it that went wrong? Is it best to try to reboot the system? How long will that take? Should a failover be attempted to the backup system? How long, in turn, will that take? Will the backup really work? When was the last time that it was tested? What is it that is going to get my boss off my back so that I can tend to the problem?

---

<sup>1</sup> *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, AuthorHouse; 2004.  
*Breaking the Availability Barrier II: Achieving Century Uptimes with Active/Active Systems*, AuthorHouse; 2007.  
*Breaking the Availability Barrier III: Active/Active Systems in Practice*, AuthorHouse; 2007.  
These rules may also be found at [http://www.gravic.com/breaking\\_the\\_availability\\_barrier\\_rules.html](http://www.gravic.com/breaking_the_availability_barrier_rules.html).

With pressures like these, it is no wonder that people make mistakes. Have you ever felt your brain turn to mush when everything seems to be going wrong? Have your fingers become too fat for the keyboard?

Times like this are when well-thought-out, tested, documented, and automated operating procedures are a must. The best way to avoid mistakes under stress is to reduce the stress. If the operating staff has a clear set of procedures that can guide them through the crisis, if they are adequately trained in those procedures, and if the procedures are highly automated, people will get “less stupider” and make fewer mistakes. As a result, your application availability will certainly improve.

**Rule 20:** *Conduct periodic simulated failures to keep the operations staff trained and to ensure that recovery procedures are current.*

This rule is really a corollary to Rule 19. The best way to reduce the effect of human errors on availability is to ensure that all involved are well-trained. Simulated periodic failures are the best way to achieve this training.

Simulated failures themselves come with a price. For one thing, the system will be down during the recovery period, whether the recovery is a reboot or a failover. For another, if recovery is to be achieved by failing over to a backup system, will that system really come online without a problem? Must the system then be failed back to the primary system once the test is over?

These problems mean that the system will be down for a good bit of the simulated failure. Is there a window during which the users of the applications running on the system can be without service? What happens if a fifteen-minute failover test turns into a four-hour outage? For these reasons, many companies do not practice recovery procedures.

A good solution to this dilemma is the use of active/active systems. It is always known that the backup system is working because it is in production. All that needs to be done is to move the affected users to the surviving system. This can often be done in seconds.

**Rule 23:** (Niehaus' Law) - *Change causes outages.*

When it comes to outages caused by people or software, a common culprit is change. Modifications to applications are a notorious source of software bugs that can cause an application outage, if not a system outage. Furthermore, changed applications can mean changes in operating procedures that can lead to operator errors.

Good change control procedures are imperative to achieve high availability. There should be sufficient software documentation so that the impact of a change on other software modules can be accurately determined. Changes should be carefully reviewed and critiqued by all affected personnel. Functional changes, software changes, and operating procedure changes should be clearly documented. Most importantly, any change should be thoroughly tested; and the operations staff should be adequately trained in any changes to operating procedures.

**Rule 24:** *Following the failure of one subsystem, failover faults cause the system to behave as if it comprises  $n-1$  remaining subsystems with decreased availability.*

A failover fault occurs when failover to a backup component is unsuccessful. It does not matter whether the backup component is a process in a checkpointed process pair or a system in an active/backup configuration.

Failover faults are insidious. First of all, it is very difficult to exhaustively test a system to ensure that it will always fail over successfully under any failure condition. That is because one cannot predict every failure scenario, and many failure scenarios cannot be easily simulated. As a consequence, the possibility of a failover fault is always present.

Secondly, failover faults have a magnified impact on system availability. For instance, it can be shown that under reasonable conditions a 1% chance of a failover fault can translate into a 40% reduction in system availability!

Again, active/active systems bring a failover fault advantage to system availability because the likelihood of a failover fault is highly unlikely. It is always known that the backup is working because it is actively processing transactions.

**Rule 26:** (The Golden Rule) - *Design your systems for fast recovery to maximize availability, to reduce the effect of failover faults, and to take full advantage of system splitting.*

No matter what the reason for a system failure, it is obvious that the faster that the system can be returned to service, the better will be the system availability. Recovery times can be improved by contracting for a higher level of service, by stocking critical parts at the data center, and by ensuring that there are enough trained technicians on call to effect parallel repair (multiple system failures are repaired simultaneously instead of one at a time).

It can be shown that reducing the average time to repair a node in a distributed system by a factor of  $k$  will reduce its average time to restore service by a factor of  $k$ , will increase its average time before failure by a factor of  $k$ , and will improve its availability by a factor of  $k^2$ .

**Rule 27:** *Rapid recovery of a system outage is not simply a matter of command line entries. It is an entire business process.*

Adding to Rule 26, system recovery is a process that encompasses the entire enterprise. It requires close cooperation and efficient decision making of the entire operating staff and their management. It requires not only that each system component has a backup but also that every critical person has a backup. There must be a means to contact important personnel on an emergency basis. It requires good operator training. It requires good documentation of the recovery procedures for the system, the applications, the database, the network, and the environmental controls; and it requires the frequent testing of these procedures. It requires a plan for continuing the business of the enterprise in the absence of IT support.

**Rule 28:** *RPO and RTO are both a function of the data replication technology used to maintain databases in synchronism.*

Synchronous replication guarantees that no committed transactions will be lost following a system failure. This is an RPO (recovery point objective) of zero. With asynchronous replication, the amount of data that may be lost is that which is in the replication pipeline at the time of failure. Thus, the lower the latency of the replication engine, the less data will be lost. Replication latency is a function of disk queuing points in the replication engine as well as process and communication queues and their service intervals.

Recovery time, which is the focus of RTO (recovery time objective), depends upon the configuration of the backup system. If the backup is a cold standby whose databases are kept current via data replication, bringing the databases into a state of consistency, loading the applications, connecting them to the databases, and testing the system before putting it into operation can take hours in some instances. If the standby system is already up and running, this time can be shortened to perhaps minutes. Using an active/active configuration can reduce recovery time to seconds.

**Rule 29:** *You can have high availability, fast performance, or low cost. Pick any two.*

The design of an active/active system brings with it many choices that lead to compromises. Virtually any decision that will increase availability, increase performance, or decrease cost will adversely affect the other attributes. A good example is the disk farm. Consider a four-node active/active system. If each node has its own complete mirrored database, this is the ultimate in availability. Performance will also be enhanced because all applications have local access to the database. However, the cost of the duplicated disk farms may be excessive.

If each node instead has an unmirrored database, availability will suffer; but cost will be significantly reduced, and performance will be unaffected. If we further restrict the system so that only two of the nodes have a copy of the database, and if each of those copies is unmirrored, cost will be further reduced at the expense of lower availability and lower performance (some nodes will now have to reach across the network to get to a database copy).

**Rule 30:** *A system that is down has zero performance, and its cost may be incalculable.*

Availability remains the dominant factor in many critical applications. Remember that a system that is down has zero performance; and the cost of that downtime may be untenable when measured not only in dollars but in customer retention, publicity, regulatory infractions, and perhaps even the loss of life or property. The cost of an active/active system is determined not by a cost objective so much as it is determined by the performance and availability requirements of the application.

## **Referential Integrity**

**Rule 34:** *Database changes must be applied to the target database in natural flow order to maintain referential integrity.*

A major factor in system recovery time following a node failure is the recovery of the database. This must be done before any applications can be run.

If a replication engine that ensures data integrity of the target database is used, this means that the only database recovery operations that need to be applied are the backing out of incomplete transactions. Database integrity implies that all related transactions must be applied to the target database in the same order that they were applied to the source database.

Any replication technique that does not guarantee referential integrity can lead to a corrupted database. In this case, the database must be repaired, which can take a time that is very much longer than that required to simply roll back uncompleted transactions.

**Rule 35:** *A serializing facility that will restore natural flow is required following all data replication threads and before the target database in order to guarantee that the database will remain consistent and uncorrupted.*

Rule 34 becomes critical in a multithreaded replication engine. Replication engines often use multiple threads to increase performance. If updates or transactions can be sent down different threads, they may arrive at the target database in an order that is different from the order in which they were applied to the source database. Therefore, some sort of serialization facility is necessary at the target database to reorder updates so that their order matches that of the source database updates.

## **Achieving Century Uptimes**

**Rule 36:** *To achieve extreme reliabilities, let it fail; but fix it fast.*

This is the mantra of active/active systems. If the recovery from a failure is so fast that users don't notice it, then in effect no failure has occurred. An active/active system can typically recover from a node failure in a time measured in subseconds to seconds. All that is required is that users on the failed node be switched over to a surviving node or that the failed node be removed from the pool of nodes that can accept transactions.

**Rule 37:** *All of the purchased capacity in an active/active system is usable. There is no need for an idle standby system.*

Every node in an active/active system is actively processing transactions. There is no spare system sitting passively by waiting to be put into service following a system failure. Even if spare capacity is built into the system to handle a failure, that capacity can be put to good use during peak times such as holidays.

**Rule 38:** *Providing that the nodes in an active/active system are geographically distributed, disaster tolerance comes for free.*

Adding to Rule 37, if the nodes in an active/active network are sufficiently geographically distributed, no single disaster will bring down multiple nodes to cause a total outage. The system is inherently disaster-tolerant.

**Rule 39:** (NonStop Maxim) - *Century uptimes in an active/active system will generally require fewer nodes if fault-tolerant nodes are used rather than high-availability nodes.*

In fact, studies have shown that using nodes each with an availability of four nines (such as NonStop servers) will typically require at least one less node in an active/active network than using nodes with an availability of three nines (such as industry-standard servers) to achieve a specified level of availability and capacity following a node failure.

**Rule 40:** (Darwin's extension to Murphy's Law) - *Eventually, a disaster will befall every enterprise; and only those that are prepared will survive.*

This should be the ultimate wakeup call for all enterprises to protect their IT assets from disaster. Consider the number of companies that were put out of business by the 9/11 terrorist attacks. Those that could continue operations even in the face of this disaster survived. Many others failed.

Disasters as serious as this are just waiting to happen, from natural occurrences such as earthquakes to man-made disasters such as biological attacks and dirty nuclear bombs.

## **What's Next?**

In our last part of this series, we will review the final excerpt of our sixty-four Rules of Availability. Taken as a whole, they represent a body of best practices for achieving high availability and even continuous availability.