



## **“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

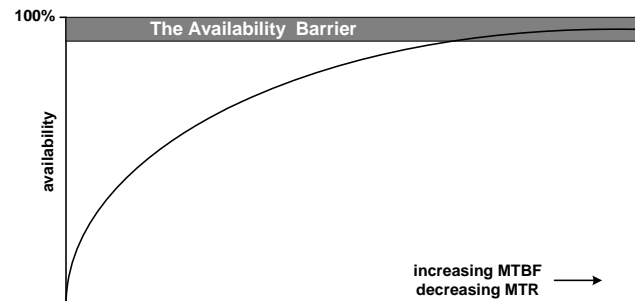
## Part 7: What is the Availability Barrier, Anyway?

November/December 2007

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

### ***The Availability Barrier is Recovery Time***

In this era of 24x7 business operations, the quest for 100% uptime seems to be continually elusive. We get better and better, but the ultimate uptime is always just a little further away. What is that barrier to 100% uptime against which we are bumping? And what can we do to break through the availability barrier?



We submit that in commercial data processing, the availability barrier is recovery time. In what follows, we will explain why this is true, and what can be done to reduce recovery time so that we can push the availability barrier back as far as possible – though it will always be there, taunting us.

### **What is Reliability?**

Let us first define what we mean by system reliability. There are three common measures of reliability – availability, or the proportion of time that a system is operational, which we denote as  $A$ ; failure interval, or uptime, which we denote as MTBF (mean time before failure); and recovery time, or downtime, which we denote as MTR (mean time to recover). These three measures are not independent. They are related by the well-known relationship<sup>1</sup>

$$A = \text{MTBF}/(\text{MTBF}+\text{MTR}) \quad (1)$$

Over the history of computers, we have made strides to increase system MTBF and to reduce its MTR and thus have made progress toward improving system availability. However, try as we might, we never achieve 100% availability; nor are we likely to. From Equation 1, 100% availability can only be achieved by either creating a system that will never fail (infinite MTBF) or by creating one that takes no time to recover from a failure (zero MTR).

As we think about it, the measure of reliability is in the eye of the beholder. Any one of these parameters may be chosen as the measure of reliability. For instance:

<sup>1</sup> W. H. Highleyman, P. J. Holenstein, B. D. Holenstein, *Breaking the Availability Barrier: Survivable Systems for Enterprise Computing*, AuthorHouse; 2003

- In a satellite system, mean time before failure, MTBF, is of paramount importance. It is critical for a system contained in a satellite to be up as long as possible (hopefully, years) because once it fails, it is generally not repairable.
- In a system measuring oil flow in a pipeline, availability,  $A$ , is the most important measure because when the system is down, oil flow is unknown. Any downtime results in revenue loss for oil delivered.
- In a commercial data processing system, we submit that mean time to recover, MTR, is the most important. We expand on this reasoning in the next section.

## The Importance of Recovery Time in Commercial Data Processing

The importance of recovery time in commercial data processing systems can best be shown by example. By a commercial data processing system, we mean a system whose users are people or other computers which submit requests and which expect a timely response.<sup>2</sup>

Assume that we have a system with an availability of four 9s (it is up 99.99% of the time). From Equation (1), it can be seen that there are any number of combinations of recovery time, MTR, and failure interval, MTBF, that lead to an availability of four 9s. Let us look at some cases:

- Case 1: The system is down one second every three hours. The users probably will not even notice this outage as it is within the response time which they expect from the system.
- Case 2: The system is down one minute every week. Users would probably be affected by the outage but would grudgingly accept it. After all, “Computers do fail.” (How often do you reboot your PC?)
- Case 3: The system is down one week every 200 years. The users would be delighted with the system until they all lost their jobs because the company went out of business. If you say “So what? The system will be gone long before then,” you are missing the point. That bad week could be next week.

Based on these examples, we submit that availability and failure intervals are not nearly as important as recovery time in these sorts of commercial data processing systems. It does not matter how much of the time a system is up or how often a failure occurs so long as *the failure interval is so short that the users don't notice it*. Therefore, it is recovery time that counts in commercial systems.

---

<sup>2</sup> More specifically, we are interested in online, real-time systems. Batch-oriented systems do not fit this argument as availability may be a more important factor.

## **The Progression of System Availability**

Availability has usually been thought of as less important than performance. Over the last several decades, while single-system performance has increased by a factor of several thousand (from one megahertz to several gigahertz), single-system availability has increased only by a hundredfold.

The search for higher availability started with efforts to increase the time between failures rather than efforts to decrease recovery times. Four decades ago, systems had availabilities of 90% (one 9). Today, due to increased MTBF, industry standard servers have availabilities of 99.9% (three 9s). Massive redundancy in fault-tolerant servers has improved this to 99.99% (four 9s). (These availabilities include all sources of failure – hardware, operating system, applications, operator error, environmental faults, and site disasters). It is now time to move our focus to recovery time.

### ***The Early Days – 90%***

When commercial computers first came on the market (in the 1950s, with Univac, IBM, and RCA), large systems built with vacuum tubes experienced significant downtime and showed availabilities of 90%. MIT's Whirlwind had an MTBF of eight hours, and many programs took longer to run than that. Check pointing interim application state to persistent storage was born out of necessity.

### ***The Commercial Renaissance – 99%***

With the advent of the transistor and its use in computing systems in the 1960s, availabilities increased by an order of magnitude to two 9s (99%). The invention of the transistor, which led to integrated circuits and the processor chips of today, was probably the most significant contributor to system reliability (and performance, as well) in the history of computing.

### ***Hardened Servers – 99.9%***

As computers became mainstream, an availability of two 9s became intolerable in many business-critical applications. System vendors began to offer systems that featured redundant components to protect against some of the more common failures. Dual power supplies and fans powered SMP (symmetric multiprocessing) systems that could survive a processor failure. RAID disk arrays that could survive the failure of any one of its disks were developed. Quality control was tightened on all components. Operating system facilities were improved to simplify system administration so as to reduce operator errors. Improved software QA testing methodologies resulted in more hardy applications.

As a result, these systems began to exhibit three 9s of availability.

## ***Fault-Tolerant Servers – 99.99%***

In the 1970s, fault-tolerant systems appeared. These systems were designed to survive any single failure. Automatic failure detection removed a failed component from service, and an equivalent component took its place nearly instantly. The operating systems were hardened to reduce software failures.

At the hardware and operating system levels, these systems proved their worth with availabilities of five 9s or more. However, application software faults, operator errors, and the necessity for planned downtime for software and hardware maintenance compromised this availability, reducing it to four 9s. Still, this was another order of magnitude improvement in availability.

It is a tribute to management short-sightedness that these systems are just now, after three decades, beginning to enter the mainstream market. Many companies are just beginning to understand the true cost of downtime and how it can be reduced by investing in fault tolerance.

## ***Moving to Multinode Application Networks***

Four 9s seemed like a major availability barrier. Up to this point, the predominant efforts described above to increase availability had been to increase the system MTBF. MTBFs of several months for standard systems and of several years for fault-tolerant systems were being measured. But it became economically infeasible and technically difficult to wring much more out of extending MTBF.

This was because the systems themselves had become so reliable that their MTBFs played an ever decreasing role in system availability. System MTBFs were being overshadowed by other causes of downtime – application software faults, operator errors, and planned downtime. When a system was lost due to one of these occurrences, it could take hours to bring it back up again. Even worse, should a disaster of some sort destroy an entire data center, it could take days or weeks to restore the data center to operation.

Clearly, the game had to shift from extending the times between the failure of systems to reducing their recovery times. The mantra became “Let it fail, but fix it fast.”<sup>3</sup> Thus was born the concept of an entire backup system that could take over processing from the primary system on short notice.

## ***Active/Standby Pairs***

The initial effort to provide a backup system involved installing a passive standby system equivalent to the primary system at some remote site. By geographically separating the primary and standby systems, protection against site and localized disasters was achieved.

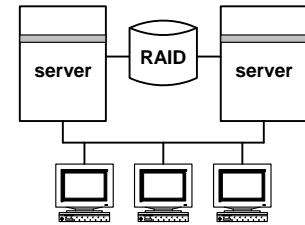
---

<sup>3</sup> See Rule 36, *Breaking the Availability Barrier: Achieving Century Uptimes with Active/Active Systems*, AuthorHouse; 2007.

However, recovery time following a primary failure was not significantly improved. In most early systems, the standby database had to be loaded from magnetic tape copies, a process that could take hours or days. Later systems used data replication techniques to maintain a copy of the database in near-real-time on the standby system. However, recovery times were still often measured in terms of hours. This architecture protected a system well from disaster, but did little to improve overall system availability.

### **Clusters – 99.999%**

About the time that fault-tolerant systems were coming onto the market, cluster technology was becoming available. A cluster comprises two or more processors with access to a local common database (typically a RAID array). One or more applications are assigned to each processor in the cluster.<sup>4</sup>



**A Cluster**

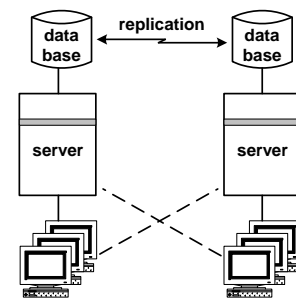
If a processor in the cluster fails, the applications which it was running are failed over to other surviving processors in the cluster. Cluster failover times today are often measured in minutes.

It is highly unlikely that more than one processor at a time will be down provided that repair is rapid. However, a new form of downtime must now be accounted for; and that is failover time. During the minutes that it might take to fail over the downed applications, the users of those applications see a downed system. The system is not available to them.

However, minutes are much better than hours. Furthermore, planned downtime can be significantly reduced by rolling upgrades through the cluster, one processor at a time. The result is that clusters can provide availabilities of more than five 9s.

### **Active/Active Systems – 99.9999%**

For many critical applications today, several minutes of downtime is unacceptable. Furthermore, clusters in and of themselves do not solve the disaster tolerance problem. An entire backup cluster must be set up at some remote site.



**An Active/Active System**

These problems are overcome by active/active systems.<sup>5</sup> An active/active system comprises two or more processing nodes in an application network, all cooperating in a common application. Each node has available to it an application database copy, which it uses to process transactions routed to it. Changes made to its database copy are replicated to the other database copies in the application network so that all processing nodes see the same database.

<sup>4</sup> Active/Active Versus Clusters, *Availability Digest*; May, 2007.

<sup>5</sup> What is Active/Active?, *Availability Digest*; October, 2006.

Should a node fail, all that is necessary is to switch users from the failed node to other surviving nodes in the application network. This switchover can be accomplished in seconds. Thus, active/active systems can reduce failover time from the minutes needed by clusters to seconds.<sup>6</sup>

Active/active systems can eliminate planned downtime by rolling upgrades through the application network, one node at a time. In addition, if the nodes are geographically dispersed, active/active systems provide disaster tolerance at no additional cost.

With failover time measured in seconds, planned downtime eliminated, and protection against site failures, active/active systems are today providing availabilities in excess of six 9s.

### **Breaking the Availability Barrier**

With the capability of returning users to service in just a few seconds, we are coming extremely close to achieving our goal of 100% availability. With six 9s availability, we have improved availability by a factor of 10,000 over the early days of computing, comparable to the performance improvements achieved over the same period of time. If we can get failover times reduced to milliseconds, users will generally never even know that a failure occurred. Failover time will have been eliminated from the availability equation. The pursuit of this goal has moved the problem from the realm of systems to the network. How do we detect a failure and fail over users in, say, 10 milliseconds?

If we can achieve failover times this brief, have we truly broken the availability barrier? Not quite. We have just moved it further back to where we are highly unlikely to ever see it. If failover is instant, there is no failover downtime. However, there is still the remote possibility that the system will go down should all of the nodes in the application network fail simultaneously. For a two-node active/active system using fault-tolerant nodes, each with an availability of four 9s, the system will have an availability of eight 9s. Assuming that it will take about four hours to return at least one node to service, this translates to a failure interval in excess of 400 centuries.

Perhaps we can rest on our laurels at this point and say that for all practical purposes, we have achieved 100% availability - until that one event that we cannot possibly foresee takes us down. Let us hope that at that time we still have an operational business continuity plan.

---

<sup>6</sup> In some active/active configurations, only one node is actively processing transactions. This configuration is known as a “sizzling-hot standby.”