# the Availability Digest

## What Is the Availability Barrier?
### March 2010

Since the dawn of commercial computing in the 1950s, the technological quest for computer engineers has been the drive to reduce the price/performance ratio of computing systems. We needed to improve their performance and at the same time reduce their cost. Through intense competition and massive investment in computing research and development, we have surpassed early expectations manyfold. Today's $400 desktop has thousands of times the processing power of the multimillion dollar mainframes of 1960.

With today's demands for 24x7 global services, we must now shift our attention to another goal – reliability. No longer is it acceptable to be down for half a day due to a processor failure or to be offline for a weekend upgrade. Our systems are expected to be up at all times. Any downtime can have a significant corporate impact that might be measured in dollars, lost customers, regulatory violations, or bad publicity.

However, the press is full of stories about catastrophic system failures. What is the barrier that we must overcome to eliminate these failures and to achieve high reliability?

First we have to understand what we mean by reliability? That depends upon what we are trying to achieve.

## What is Reliability?

There are three measures of system reliability:

- *Availability*, or the proportion of time that a system is operational. Availability is represented by the symbol "A" for "availability."

- *Failure Interval*, or the average time between system failures. Failure interval is represented by the symbol "MTBF" for "mean time between failures."

- *Recovery time*, or the average time that it takes to return the system to service following a failure. Recovery time is represented by the symbol "MTR" for "mean time to recover."

These three parameters are related by the well-known relationship:

$$A = \frac{MTBF}{(MTBF + MTR)} \tag{1}$$

100%    **The Availability Barrier**

availability

increasing MTBF
decreasing MTR
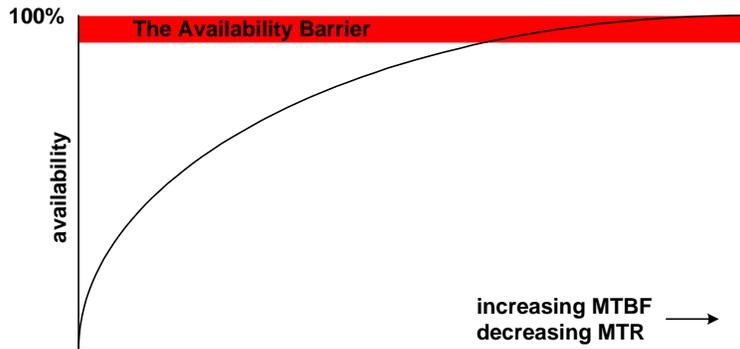
Over the last decades, we have attempted to improve reliability by increasing MTBF, the time between system failures. We have become quite successful at this. In the early days of vacuum-tube computing, the Whirlwind computer at MIT had an MTBF of about eight hours. Today, computers might fail once or twice per year.

But try as we might to reduce our failure rates, we will never achieve 100% availability. From Equation 1, continuous availability requires either that we have an infinite MTBF or a zero MTR. Neither is practically achievable.

So which of these parameters, A, MTBF or MTR, should we concentrate on to improve reliability? As it turns out, reliability is in the eye of the beholder. For instance:

- In a satellite, the failure interval, MTBF, is of paramount importance. Systems in a satellite are typically not repairable. Therefore, when a system fails and cannot be recovered by commands from the ground, the satellite fails. There is no recovery time, and availability is meaningless. The satellite is either operational or it has died.

- In a system measuring oil flow in a pipeline, availability, A, is the key parameter. To the extent that the system is down, no oil flow is recorded; and revenue for oil delivered is lost.

- We submit that in commercial data-processing systems, the recovery time, MTR, is the important parameter. We expand on our reasoning next.

## Recovery Time – The Availability Barrier for Commercial Data Processing

By commercial data-processing systems, we mean those that run applications in which users are people or other computers that submit requests and that expect a timely response. These systems are typically running OLTP (online transaction processing) applications that are often mission-critical to the company. We do not include batch systems in this discussion as availability (A) may be a more important parameter to batch systems than recovery time (MTR).

The importance of recovery time in critical applications can be illustrated by some examples. Assume that you have an application with three nines of availability. Which set of reliability parameters do you think users would prefer?

- Case 1: The application is down one second every sixteen minutes. The users will probably not even notice these outages if they are within the expected response time of the application.

- Case 2: The application is down one hour every six weeks. Users are likely to complain but grudgingly accept it. After all, "computers do fail," they may say.

- Case 3: The application is down one day every three years. If the application is mission-critical, this could have a serious impact on the company, perhaps causing it to halt operations or to revert to manual procedures at significant cost or peril to the company.

We submit that Case 1 is acceptable, Case 2 is painful, and Case 3 in unacceptable. The interesting thing about these cases is that they all have the same availability of three 9s; so availability, A, is not a factor in acceptable reliability for commercial systems. Even more to the point, the failure interval, MTBF, is shorter (worse) for the better cases; so MTBF is certainly not the driving factor.

The driving factor is recovery time, MTR. In other words, *let it fail, but fix it fast*. This is the reliability mantra for critical commercial data-processing systems. In fact, if recovery is so fast that no one notices that there has been an outage, continuous availability has effectively been achieved.

## The Progression of System Availability

Up until recently, availability has been a second cousin to performance. Over the last sixty years, performance has increased by a factor of several thousand, from one million instructions per second to several billion instructions per second. But the availability of nonredundant systems has only increased by a hundredfold, from one 9 to three 9s.

To illustrate this point, let us look at the history of availability:

### The Vacuum-Tube Days – One 9

Computing in the 1950s was done by vacuum-tube monsters. These systems had to be taken down every week or so for preventive maintenance. For instance, operating voltages were reduced to see which vacuum tubes would fail; and they were replaced to try to minimize unplanned failures during normal operations.

### The Solid-State Renaissance – Two 9s

The advent of the transistor, which led to integrated circuits and the powerful processor chips of today, was probably the most important contributor to availability (and performance) in the history of computing. Solid-state systems in the 1960s showed an order of magnitude more availability than their vacuum-tube predecessors.

### Hardened Servers – Three 9s

As we entered the 1970s, two 9s of availability (80 hours of downtime per year) became intolerable for many critical applications. To reduce the frequency of failures, system vendors began to include redundancy in critical components that were subject to the most failures. This included power supplies, fans, RAID disk arrays and SMP (symmetric multiprocessing) processors. On the software side, operating-system facilities were improved to ease system management so as to reduce operator errors.

### Fault-Tolerant Servers – Four 9s

By the late 1970s, it was becoming clear that further efforts to increase the MTBF of a single nonredundant system were having less and less impact. Components were going to fail, and nothing could be done about that. This realization led to the emergence of fault-tolerant systems

such as those from Tandem and Stratus (and many others that did not survive). In these systems, faulty components were automatically detected. They were removed from service and replaced virtually instantly with an equivalent component. The operating systems were hardened to reduce software failures. In Tandem systems, a failure caused by a software bug could often be recovered by switching to another copy of the application that had been kept synchronized with the active copy.

### Increasing MTBF – The End of the Line

Fault-tolerant systems in and of themselves achieved availabilities of five nines. However, other failure reasons, such as manual errors and software bugs, reduced this availability to four nines. What was needed was a way to quickly switch to another system if the active system failed. The concept of "let it fail, but fix it fast" took a major step forward. No longer was it acceptable to simply replace failed components. Many failure modes required that the entire system be replaced – and fast. However, the existing technology for recovering from a massive system failure – magnetic tape backup to a standby system – was woefully inadequate for high availability since failover took hours or days.

### Clusters – Five 9s

Clusters solved the failover time problem to a great extent. A cluster comprises two or more processors with access to a common database. Only one processor can be updating the database, but the others are standing by in case the active processor fails. With clusters, failover can be in minutes, leading to availabilities in the order of five 9s (five minutes of downtime per year). This was the first step in achieving very high availability by rapidly failing over to a backup system.

### Active/Active Systems – Six 9s and Beyond

For many critical applications, several minutes of downtime are simply unacceptable. Besides, clusters cannot be geographically separated; and a site disaster can took down the entire cluster. Recovery has to be made to a remote backup system, which even with modern-day recovery procedures such as virtual tape and database replication can take hours. To go beyond five 9s, systems must be protected against site disasters as well as against system faults; and recovery times must be reduced from minutes to seconds.

Active/active systems solve this problem.[1] The technology for active/active systems appeared in the 1990s when bidirectional data-replication engines were introduced.[2] An active/active system is a network of geographically-distributed processing nodes, each having access to a distributed copy of the application database. The database copies are kept in synchronism via data replication. All processing nodes are actively engaged in the application, and a transaction may be sent to any node in the application network. Should a processing node fail, all that is required is to reroute transactions to surviving nodes. Failover is measured in subseconds to seconds.

Furthermore, active/active systems eliminate planned downtime. Upgrades can be rolled through the nodes one at a time by taking each node down, rerouting traffic to the other nodes, upgrading the downed node, and returning it to service.

## Breaking the Availability Barrier

---

[1] What is Active/Active?, *Availability Digest*; October 2006.
[2] A limited form of active/active systems was introduced by Digital (now HP OpenVMS) in 1984. Its split-site clusters were truly active/active systems, but the synchronous replication technology that these systems used limited the geographical separation of the data centers. See OpenVMS Active/Active Split-Site Clusters, *Availability Digest*; June 2008.

It took decades for computer technology to progress to the point that system reliability could no longer be significantly improved by increasing failure intervals. To achieve six 9s and beyond, we must now look to recovering rapidly from the failures that we know are bound to occur. The technology is now here to provide failover times measured in seconds or even subseconds.[3] We break the availability barrier by focusing on MTR, the system recovery time, rather than on MTBF, the failure interval. Let it fail, but fix it fast.

The availability barrier is recovery time. But even if we can reduce recovery time in an active/active system to milliseconds, have we really broken the availability barrier? Have we achieved absolute continuous availability? Not quite. We have moved the availability barrier back, but we haven't eliminated it. There is always the remote possibility that all nodes in the system will fail. In this case, we may be down for hours before we can restore enough nodes to service. For instance, if a pair of fault-tolerant nodes is used in an active system, each with an availability of four 9s, the system will have an availability of eight 9s. If recovery time from a total node failure is four hours, this translates to one four-hour failure every 400 centuries.

Why worry about that? Won't we be long gone by then? Maybe and maybe not. That one failure in 400 centuries might just happen tomorrow. Let's make sure that if this unlikely failure should occur, we still have an operational business-continuity plan.

---

[3] Achieving Fast Failover in Active/Active Systems – Parts 1 and 2; *Availability Digest*; August, September 2009.