"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
17 General Warren Blvd.
Malvern, PA 19355
610-647-6250
www.ShadowbaseSoftware.com

Achieving Century Uptimes

Part 19: Reviewing Three Years of The Availability Corner

November/December 2009

Dr. Bill Highleyman (editor@availabilitydigest.com)
Paul J. Holenstein (shadowbase@gravic.com)
Dr. Bruce D. Holenstein (shadowbase@gravic.com)

Happy Anniversary! We have just passed the three-year mark in publishing our continuous availability column, "The Availability Corner: Achieving Century Uptimes," in *The Connection*.

The Availability Corner focuses on high availability and continuous availability – the ability of a system to provide near-continuous user services 24x7 with little or no downtime. Can we really achieve zero downtime? Of course not. But if the recovery from an outage is so short that no one notices, then in effect there has been no outage. This is the secret of active/active systems – *let it fail, but fix it fast*.

Recognizing that some reading this column may not have seen our earlier articles, we thought it appropriate to briefly review the eighteen columns that have been published since this series first debuted in the November/December 2006 issue of *The Connection*. We also would like to map out some of the future topics that we plan to address in upcoming issues.

Part 1: Survivable Systems for Enterprise Computing (November/December 2006)

We began our series with a description of active/active technology. An active/active system is a network of peer processing nodes, each having access to a common distributed database. Any node can execute any transaction. The distributed database copies are synchronized via data replication. Should a node fail, all that is required for recovery is to reroute users to surviving nodes. We pointed out that even with such a highly reliable system, a business-continuity plan is still required. After all, that one failure in a hundred years might happen tomorrow.

Part 2: What Will Active/Active Cost Me (January/February 2007)

Estimating the cost of an active/active system can be complex because there is great flexibility in how it can be implemented – the number of processing nodes and database copies, the degree of network redundancy, the geographically distributed sites, licensing, and so on. The first step is to carefully define the availability requirements for each application. The system can then be configured to meet these availability requirements. The other side of the equation is the cost of downtime against which these costs must be justified.

Part 3: Avoiding Data Collisions (March/April 2007)

A major problem in designing active/active systems is in avoiding data collisions if asynchronous replication is used to maintain database copies in synchronization. A data collision occurs if the same data item is changed at the same time in different database copies. In this two-part article, we talk about various architectures and techniques that can be used to avoid data collisions. Included are application characteristics, database partitioning, and hierarchical node configurations.

Part 4: Resolving Data Collisions (May/June 2007)

We continued our discussion of data collisions with techniques for resolving data collisions that cannot be avoided. We showed how to estimate the collision rate and how to detect collisions when they do occur. Once detected, a collision must be resolved. One powerful technique is relative replication, in which operations rather than changed data are replicated. Alternatively, the winner of a data collision can be determined by data content, node precedence, fuzzy replication, or special business rules. If all else fails, collisions can be resolved manually.

Part 5: Modular Redundancy – To Need or Not to Need (July/August 2007)

At about this time, HP announced its Neoview database appliance. Though based on NonStop technology, Neoview took an interesting alternate approach. All NonStop K-series, S-series, and Integrity servers use lock-stepped processors. A disagreement takes the processor offline, thus preventing the propagation of errors and the corruption of the database. Neoview dropped this protective feature. Was this change warranted? This article explores the rationale behind this decision and its consequences.

Part 6: Active/Active versus Clusters (September/October 2007)

Clusters are the traditional approach to achieving high availabilities of five 9s or so. How do they compare to active/active systems with availabilities of six 9s and beyond? Failover within a cluster takes minutes rather than the seconds required for an active/active failover. Clusters must use collocated nodes and are not disaster-tolerant, whereas active/active systems are inherently disaster-tolerant. Clusters are not as easily scalable as active/active systems. However, cluster technology is more mature than that of the more recent active/active upstarts.

Part 7: What is the Availability Barrier Anyway? (November/December 2007)

“Reliability” is defined in many ways. In transaction-processing systems, reliability is a measure of how often and for how long users are inconvenienced. We argue in this article that reliability is measured by recovery time. We show that over the years, great advances have been made in system failure intervals. But recovery times continue to be measured in hours or at best minutes, which is unacceptable in many applications. Active/active systems overcome this barrier with recovery times so fast that users often don’t realize that an outage has occurred.

Part 8: Let's Make Availability a Part of Performance Benchmarking (January/February 2008)

Performance benchmarks have become an accepted method for comparing systems. But system availability often seems to be an afterthought. With today's high cost of downtime, availability can be as important as performance in determining TCO (total cost of ownership). But how can we get reasonable availability measures of systems that seldom fail? In this article, we recognize that most failures in today's systems are caused by operator, software, and environmental faults. By including the cost of downtime, we propose an availability benchmark that is simple to use.

Part 9: Where Is My Database of Record? (March/April 2008)

An active/active system comprises two or more synchronized copies of the application database. But in many cases, for regulatory or corporate reasons, one of these copies must be designated the "single version of the truth" – the database of record. The choice of the database copy to be the database of record is influenced by many factors. If the nodes in the system are arranged hierarchically, the highest-precedence node should contain the database of record. In a symmetric system, the choice can be made based on administrative or maintenance support.

Parts 10, 11, 12: The Rules of Availability I, II, III (May/June 2008, July/August 2008, September/October 2008)

This three-part series summarized the availability rules that were postulated in the three-volume series, *Breaking the Availability Barrier*.[1] Pertinent rules, including some counter-intuitive favorites of ours, follow:

Rule 2: Providing a backup doubles the 9s.

Rule 3: Adding processors to a system makes it less reliable.

Rule 6: System availability increases dramatically with increased sparing. Each level of sparing adds a subsystem's worth of 9s to the overall system availability.

Rule 11: Minimize data-replication latency to minimize data loss following a node failure.

Rule 12: Database changes generally must be applied to the target database in natural-flow order to prevent database corruption.

Rule 15: Minimize replication latency to minimize data collisions.

Rule 19: When things go wrong, people get stupider.

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

Rule 23: Change causes outages.

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

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

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

---

[1] W. H. Highleyman, P. J. Holenstein, B. D. Holenstein, *Breaking the Availability Barrier: Volumes 1, II, III,* AuthorHouse; 2004 and 2007.

Rule 40: Eventually, a disaster will befall every enterprise; and only those that are prepared will survive.

Rule 41: Active/active systems can provide the availability of a primary/standby pair with less equipment and less cost.

Rule 53: ZDM (zero downtime migration) eliminates planned application downtime, thereby improving application availability.

Part 13: Synchronous Replication: Pros, Cons, and Myths (November/December 2008)

Synchronous replication eliminates data collisions as well as data loss following a source-node failure. However, it slows down applications as they wait for transactions to be committed across the network. This article explains the advantages and disadvantages of synchronous replication as compared to asynchronous replication. Synchronous replication can limit the degree of disaster tolerance that can be achieved since internodal distance may be limited. However, recovery characteristics and server throughput are not affected.

Part 14: The Evolution of Real-Time Business Intelligence (January/February 2009)

Real-time business intelligence makes available the data related to events as they happen. It can significantly improve customer service and other enterprise activities. This article reviews the path that has been taken to get to real-time business intelligence, from data warehouses to data marts, enterprise application integration (EAI), and event-driven operational business intelligence. Impediments to real-time business intelligence are data latency (how stale is the data) and data unavailability. Active/active systems can be part of a cure for both of these problems.

Part 15: Zero-Downtime Migrations for Active/Active Systems (March/April 2009)

If a system must be continuously available 24 hours a day, seven days a week, taking it down for maintenance or upgrades is not an option. The elimination of such planned downtime requires the availability of one or more redundant systems with fast and reliable failover. Active/active systems provide this capability. To upgrade a node, users are simply moved to another node in the application network; the node is taken offline and upgraded; and it is then returned to service. The upgrade can be rolled through the application network in this manner.

Part 16: Zero-Downtime Migrations for Active/Backup Configurations (May/June 2009)

The concept of zero-downtime migration discussed in Part 15 is carried forward to active/backup system configurations in Part 16. Typically, if a backup system is available, a failover to the backup is made so that the primary system can be upgraded. However, such a failover can take hours and is often very risky. Using the data-replication techniques perfected for active/active systems, this article explains how users can be switched to a backup system quickly and reliably so that the primary system may be upgraded with no user downtime.

Part 17: HP Unveils Its Synchronous Replication API for TMF (July/August 2009)

Though there are several asynchronous replication engines available for NonStop systems, there had been no synchronous replication engines because there was no way to include them in a TMF transaction. To cure this, HP has introduced its Synchronous Replication Gateway (SRG) for TMF. This article describes the SRG architecture and its use in a *coordinated-commit* synchronous replication engine. The coordinated-commit protocol utilizes asynchronous replication to replicate database updates and synchronizes with TMF only at commit time.

Part 18 – Recovering from Synchronous Replication Failures (September/October 2009)

The recovery from a failure of synchronous replication is more complex than it is for asynchronous replication. During a failure, changes to the failed node are queued. When the failed node is returned to service, the queued changes are first drained asynchronously to resynchronize the failed node. When the queue has become *small enough*, synchronous replication is begun. However, this means that there is a time during which asynchronous and synchronous replication are occurring simultaneously. This must be carefully managed.

What's Next?

The Availability Corner continues. In the future, we expect to publish articles on the following topics:

- Comparing and repairing databases in an active/active system. We would like to think that the high-availability and continuous-availability environments we have built work perfectly, but there are a variety of reasons why databases might get out of synchronization. We will explore the techniques to check for and correct discrepancies.

- How to pick a data-replication solution. There are many ways in which data replication can be used to improve the availability of a system as well as to achieve several other advantages in system operation.

- The myriad advantages of active/active systems. Besides extraordinary availability, there are many other advantages of active/active systems. They include risk-free failover testing, lights-out operation, and easy capacity expansion and load balancing.

- Achieving fast failover. Active/active systems achieve continuous availability by being able to failover users to a surviving system in seconds or subseconds. How do you do this? We will talk about user redirection, network redirection, and server redirection.

- Transitioning from a primary/backup configuration to an active/active system. You have two nodes now, but one is idle or doing other work until needed. Why not put it into service in an active/active configuration? What does it take to make this step?

- Managing an active/active system. The management of a distributed system brings many more challenges that are not faced by a single system. In addition to extended networks, multiple nodes and multiple copies of each application must be monitored.

- Third-party product areas important to active/active. We have talked mainly about data-replication products; but areas such as security, management, and monitoring are equally important. Third-party applications may also need to be modified to run active/active.

Please let us know what you think of the article series and if there are additional topics related to availability that you would like covered by emailing the authors at the addresses given above.