the Availability Digest

Continuous Availability Systems Design Guide

January 2007

The Continuous Availability Systems Design Guide[1] is an excellent coverage of issues that confront an organization interested in moving from a classical computing environment to a continuous availability environment.

It is not a “best practices” book for ensuring continuous availability, though it does touch on many of these topics. Rather, it deals with the decision-making process in determining how to approach the design and evaluation of a continuous availability solution.

Though dated (it was published in 1998), its approach is still current. In fact, even though the term “active/active” was not even coined then, it deals frequently with active/active solutions.

What is Continuous Availability?

This book defines three levels of availability:

> *High Availability* means that the system is operational during its intended hours of service. High availability is important if the cost of downtime is severe.
>
> *Continuous Processing* means that the system provides services full-time 24 hours per day, 365 days a year.
>
> *Continuous Availability* is high availability plus continuous processing. The system provides services all of the time with a high degree of availability.

Continuous availability generally implies that there are redundant systems and that outages are short enough so as not to be seen as outages by the users.

There is a definitive compromise between cost, recovery speed, and outage coverage. Improving any one of these will have a negative impact on one or both of the others.

Six activities required for the planning and implementation of a continuous recovery capability are described:

- Determine what the business requires.
- Define the data processing requirements.
- Design the continuous availability solution:
    - o System

[1] *Continuous Availability Systems Design Guide*, ITSO Redbooks SG24-2085; 1998.

  - Applications
- Select products to match the design.
- Implement the continuous availability solution.
- Keep the solution up-to-date.

The book is less an exercise in system design than it is a comprehensive discussion of planning considerations pertaining to continuous availability. Continuous availability is a corporate issue, not a data processing issue. As such, it requires

- a major investment.
- an analysis of business process priorities and importance.
- major changes in the corporate data processing culture.
- an acceptance of the residual risks not covered by the solution.
- above all, a management commitment.

Determine What the Business Requires

Continuous availability brings with it two major advantages:

- It reduces the impact of downtime since outages are rare and brief.

- It allows extended periods of service.

A set of service level objectives must be established for each business process. These service level objectives document, among other things,

- the required hours of service.
- the maximum number of outages allowed per time interval.
- the maximum allowable recovery time from an outage.

The maximum number of outages per time interval defines the average time between outages. If this time is too short, critical functions such as batch processing might be frequently interrupted.

The maximum allowable recovery time is important since an extended recovery time would deny users service during this time and could seriously impact the business.

Determine the Data Processing Requirements

Now that service level objectives have been defined, it is important to understand why the current system is not meeting these requirements.

The first step is to understand the availability performance of the current system. This is accomplished via a Past Outages Analysis. This analysis uses whatever data is available to determine the reasons for periods of past unavailability (hardware, software, etc.), whether each outage was avoidable (such as an operator error), the root cause of each outage (i.e., that problem which, if corrected, would have prevented the outage), and what the recovery procedures were.

This analysis leads to a Component Failure Impact Analysis. For each component in the system (such as processors, disk subsystems, channels, networks, workstations), a determination is made of the impact of that component's failure on the system operation. Will it take down the entire system, will it affect only a set of users or functions, or will it have no impact (such as a redundant component). The method for detecting each component failure, the failure detection time, and the failure frequency is also documented.

In addition to unplanned outages, there are requirements for planned downtime. These include batch processing, database backups, database reorganization, and replication of the database to other systems. These must be determined and documented.

With this current operational data available, the service level objectives determined in the previous step must now be converted to data processing availability requirements. Providing a level of availability to a business process means imposing corresponding availability requirements on the data processing components – the applications and data sets - that support it. Thus, the availability of each application and data set can be specified:

- Its hours of operation.
- Its availability (its percentage of uptime during its service time).
- Its maximum acceptable downtime.

If a component is shared by several business processes, it must support the availability requirements of the most critical business process.

The sum of this information is called the *application inventory*. Based on this inventory, a Service Level Agreement (SLA) can be structured. This will be the defining document for the design of the continuous availability solution.

Design the Continuous Availability Solution

The bulk of this book deals with the design of a continuously available system.

Redundancy

The heart of a continuously available system is redundancy. If failover is to be virtually instantaneous so far as the user is concerned, there must be another system that is immediately available to take over the role of a failed system.

Environment

Environmental factors include power, cooling, building facilities, and network access.

The primary power source should be backed by an uninterrupted power system (UPS) or be available from independent suppliers.

The cooling system should be redundant, with each side powered from an independent source. Automatic temperature and humidity monitoring must be provided.

The building facilities could be damaged by fire, earthquake, aircraft impact, flood, malicious acts, or social unrest (riots). In addition, the building might become unavailable for an extended period of time due to power, water, or other maintenance activities. Alternate facilities must be planned.[2]

Access to the external network should be accessible from two different points, with a fast, non-disruptive switchover.

[2] See *Fire in the Computer Room, What Now?*, ITSO Redbooks, SG24-4211; 1997.

Hardware

Processors – Single processors can contain a great deal of redundancy, ranging from dual power supplies to symmetric multiprocessors accessing a common memory. However, ideally two or more processors will be a provided in a redundant configuration. They could be arranged in an active/backup configuration or in an active/active configuration.

Channels – There should be at least two I/O channels connecting each storage device and any other peripherals that are critical to system operation.

Storage – RAID (random arrays of independent disks) is an economical and common way to provide storage redundancy. A RAID array comprises a set of disks with one or more disks than are needed to provide the desired capacity. Data is striped across the disks in such a way that it can be recreated even in the event of a disk failure. There are several configurations of RAID arrays, from RAID 1 to RAID 6. RAID 3 and RAID 5 are the ones in common use today. RAID 6 provides protection against dual disk failures.

Network

A redundant network backbone should be provided so that communication paths can be rerouted around failed components. Each host should have multiple gateways to the network. For local area networks, each user should have redundant attach points.

Operating System

An operating system can fail for several reasons, including hardware failure, corrupted files, or programming error. If the system cannot provide a fast reload following such a failure, the only protection against an operating system failure is to have a backup standing by, ready to take over.

The backup can be put to good use for operating system patches and upgrades. New patches and upgrades can be applied to the backup system first and thoroughly tested. The roles of the active and backup systems can then be interchanged until the next patch or upgrade is to be installed.

Applications

Recovery from application failures can be done by running the applications in an active/active or active/hot standby configuration. If the application is well tested, failures are typically due to an unusual sequence of events or to race conditions. It is unlikely that the same error will occur at the same time in multiple instances of the application, so routing subsequent transactions to an alternate instance should allow continued system operation.

If the application is designed for fast restart, it may be satisfactory to simply restart it.

Operational Focal Point

The operational focal point is the area in which all system maintenance consoles and system operators are located. There must be provision to replace a failed console or to allow its functions to be taken over by another console.

Integration and Isolation

An integrated system is one in which a small number of well-managed, well-maintained systems provide application services. A small number of large systems will have failure rates less than those of a large number of smaller systems.[3]

Isolation is achieved by running different applications in their own space so that they cannot affect or corrupt other applications. Isolation can be obtained by running applications in their own systems. Isolation can also be obtained on a single system by providing private address space, multiple operating system images, and dedicated disk pools.

Active/active systems provide both integration and isolation.

Automation

Operator errors comprise a major source of system failures. Therefore, it is important to automate as much of the system operations as possible. This includes active system monitoring, automatic fault recovery, network recovery, and failover to a backup system.

To the extent possible, there should be one focal point for multiple systems.

Concurrent Maintenance and Repair

Scheduled downtime must be minimized. The system must be maintainable while it is still running. This includes hardware repair and reconfiguration, operating system upgrades, and subsystem maintenance.

System changes must be thoroughly tested before being put into service. However, one must be prepared to fall back to a known working configuration if necessary.

Design Applications Towards Continuous Availability

Application Characteristics

Applications suitable for continuous availability must have the following characteristics:

- *Correctness* – It is error free and performs the specified functions.
- *Robustness* – It handles unanticipated events.
- *Extendibility* – It is easily extended to new functions.
- *Reusability* – Proven correct software can be used in new applications.

Design for Fault Tolerance

Applications should be designed so that they can determine that a problem has occurred, can isolate the failed component, can report the problem, and can quickly recover from or bypass the problem.

[3] See Can 10,000 Chickens Replace Your Tractor?, *Availability Digest*; December, 2006.

Design for Failure Resistance

Isolation is the key to failure resistance. The application should be isolated from any problems experienced by other applications. Applications should communicate with each other via messaging to ensure isolation.

Data isolation is equally necessary. Using logical database definitions and partitioning are ways to provide data isolation. The users should be able to provide a subset of services should access to part of the database be compromised.

Finally, recovery from any failure should be fast.

Availability Management

A comprehensive application monitoring and failure reporting system must be provided. Transaction volumes and response times should be measured to anticipate growing problems. The impact of outages should be recorded. Audit trails should be maintained to improve the recovery of data.

Design for Nondisruptive Changes and Maintenance

Adding or changing users, terminals, or global parameters should be possible without system interruption. To provide nondisruptive maintenance, the system should allow coexistence of different versions of applications. Providing a backup system or an active/active configuration yields an environment conducive to testing and deploying new application versions.

Design for Continuous Applications

Planned outages should be eliminated. This includes batch processing and database backup, reorganization, and replication.

All critical data must be stored in persistent storage such as disk. This storage should be redundant, such as RAID arrays. Auditing of data changes should be provided to facilitate data recovery to a known and consistent state.

Systems Management

A problem management facility should be provided to recognize, report, and resolve or bypass problems.

Changes are essential for the stability of the system. A change management facility should collect, prioritize, and schedule changes. Provisions must be made to test changes and to fall back to a known working configuration if necessary.

Availability must be managed as described above and evaluated against the Service Level Agreement.

Select Products to Match the Design

Once the design has been completed (or even as the design proceeds), products must be chosen to implement certain of the system requirements. For those requirements for which no product exists, in-house implementation is required.

Products should be chosen first based on the required functionality, then on product quality. Product quality may be inferred from the reputation of the supplier, the support level provided, and the maturity of the product.

Once products have been chosen and the in-house development requirements understood, an estimated cost for the solution can be determined. This should include hardware, software, networks, services, implementation, and maintenance.

This cost can now be compared to the perceived advantages of a continuous availability solution. These benefits are primarily the reduced cost of downtime and the benefits of extended service hours.

Implement the Continuous Availability Solution

The hardware and software products and in-house implementations must be installed, customized and tested. Automation, recovery, rollback, and failover procedures must be developed.

Recovery procedures start with the isolation of a failed component and continue with the replacement or repair of that component. The component and perhaps the affected subsystem must be restarted and returned to service.

Redundant processors can be configured as an active/backup pair, as an active/active system, or as a duplicate-processing pair in which both processors perform identical processing functions and compare their outputs.

Keep the Solution Up-to-Date

The continuous availability solution must be maintained as new applications are installed, the hardware is reconfigured, the network changes, the organizational structure changes, and so on. Change management is the key to this activity.

The system procedures must be continually tested to ensure that they are up-to-date and to keep the staff trained. Test scenarios should include processor workload reallocation, disk volume relocation, disk volume reconstruction, processor reconfiguration, RAID or other disk recovery, channel path switching, and network reconfiguration.

Summary

The Continuous Availability Systems Design Guide is written from the mainframe user's viewpoint. As such, it is a rather complete description of the issues which a mainframe shop will face should they desire to move to continuous availability. However, the issues are applicable to any data processing organization seeking to improve its availability, whether it be an industry-standard server shop or one using fault-tolerant computers.

Written by IBM, the Guide includes an appendix which describes many of IBM's continuous availability products.