Designing Highly Resilient Systems for the Digital Age
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>3</td>
</tr>
<tr>
<td>The Critical “IT News Event”</td>
<td>3</td>
</tr>
<tr>
<td>Resilient IT Solutions and Understanding Failures</td>
<td>4</td>
</tr>
<tr>
<td>Types of Failures</td>
<td>4</td>
</tr>
<tr>
<td>Avoidable Failures</td>
<td>4</td>
</tr>
<tr>
<td>Masked Failures</td>
<td>5</td>
</tr>
<tr>
<td>Hard Failures</td>
<td>7</td>
</tr>
<tr>
<td>Soft Failures</td>
<td>7</td>
</tr>
<tr>
<td>Tools Used to Detect Anomalies</td>
<td>8</td>
</tr>
<tr>
<td>IBM® Z® Operations Analytics</td>
<td>8</td>
</tr>
<tr>
<td>Predictive Detection of Soft Failures</td>
<td>9</td>
</tr>
<tr>
<td>Runtime Diagnostics</td>
<td>9</td>
</tr>
<tr>
<td>Handling System Failure Conditions</td>
<td>10</td>
</tr>
<tr>
<td>Actions to Take If a Failure Occurs</td>
<td>10</td>
</tr>
<tr>
<td>Disaster Recovery</td>
<td>12</td>
</tr>
<tr>
<td>It is Critical to Separate Planning for High Availability vs. Disaster Recovery</td>
<td>12</td>
</tr>
<tr>
<td>GDPS Delivers Fully-Automated, Multi-Site Disaster Recovery</td>
<td>13</td>
</tr>
<tr>
<td>Cyber-Resiliency and Data Corruption Detection</td>
<td>14</td>
</tr>
<tr>
<td>Logical Corruption Protection</td>
<td>14</td>
</tr>
<tr>
<td>Engineering Resiliency into the Solution</td>
<td>14</td>
</tr>
<tr>
<td>Validating a Resilient Solution</td>
<td>14</td>
</tr>
<tr>
<td>Conclusion</td>
<td>15</td>
</tr>
<tr>
<td>References</td>
<td>16</td>
</tr>
</tbody>
</table>
I. Introduction

Over time, IT systems have continued to become more reliable. However, when large problems or IT failures do occur, the collateral damage can be staggering. With today’s demands for always-on systems, there is a need to move beyond highly available and reliable systems to highly resilient systems; that is, systems able to survive failures. A resilient system can isolate, remove, tolerate and recover from failures in hardware, software, storage, network and other components so that application and business impact are minimized. Based on IBM’s decades of experience with enterprise class IT solutions, a resilient solution requires appropriate capabilities to handle different types of failures, including building reliability into the core design.

This white paper summarizes IBM’s experience around building resilient IBM Z® systems and is aimed at the IT Management team responsible for defining and operating large commercial IT environments needing superior qualities of service. The white paper organizes and summarizes IBM’s experience and discusses the following topics, with an emphasis on z/OS®-based systems including capabilities of products and practices that are designed to help improve resiliency

- Resilient IT Solutions and Understanding Different Types of Failures
- Tools Used to Detect Anomalies
- Handling System Failure Conditions
- Engineering Resiliency into the Solution

The Critical “IT News Event”

From time to time a news event or nowadays, a social media broadcast, makes public the effects of an IT outage at a major company, which can result in angry customers, upset stakeholders, brand erosion, and even regulatory penalties. With IT providing critical solutions for customers around the world, IT solutions must be designed to avoid events of such proportions.

Resilient systems should be able to detect and recover from system issues, thus avoiding such external publicity wherever possible. To most stakeholders, it doesn’t matter what caused an event, only that the event occurred and is resolved. To prevent or contain such system issues, customers need a resilient solution, a solution that has the ability to absorb failure(s) or contain damage while continuing to deliver service. Over time, making an assumption that systems will incur some level of failure, the ability to respond and recover from failures seamlessly becomes a vital area of concern.

It is the hardening and resiliency of the system that describes its ability to provide service despite problems or catastrophic errors.
II. Resilient IT Solutions and Understanding Failures

A review of problems reported to IBM based on problem reporting records indicates that many enterprise-class IT solutions actually require a combination of events to cause a problem that is severe enough to generate such publicized outages. A resilient IT solution needs to contain the effect of unexpected events to prevent disruption, an especially challenging task when events occur in close succession. It needs to recover from combinations of events, to isolate problems, to minimize impact, and to gather diagnostics to prevent a recurrence. We outline below some key insights and examples of the approaches used by IBM in building resiliency into its solutions and can serve as an example of what customers can do to fortify their IT solutions. We discuss:

- A way of describing failures to help define what is needed for a resilient solution
- Examples of what the solution needs to do if and when a failure occurs
- Examples of how z/OS uses engineering techniques designed to make z/OS more resilient

Types of Failures

If a failure occurs, it can cause additional problems that can increase the impact of the event, often referred to as sympathy sickness. In these situations, the failure can cascade, causing additional failures to other parts of the system. A key to building a resilient solution is to understand the types of unexpected failures IT may face, and to implement controls to prevent and remediate such failures. We categorize the failures as follows:

- **Avoidable failures** that can be prevented using the capability of solution components, such as the capabilities of an operating system like z/OS.
- **Masked failures** where a component (including application code) detects a problem, corrects or mitigates the problem, and successfully hides the problem from users.
- **Hard failures** where a solution component detects a problem and fails immediately, or when the component’s recovery is unable to successfully complete, resulting in the need for another component to take over.
- **Soft failures** where the users of the solution observe a problem that is not detected by any of the solution components, or when components don’t fail, but simply perform poorly. These “sick, but not dead” incidents can be particularly difficult to diagnose and may require advanced software techniques to detect them.

Avoidable Failures

Avoidable failures can be caused by overloading a system, incorrect configuration of a system or component, operational errors during routine corrective operations, or during system changes.

Common failures include those caused by extreme spikes in usage, less than optimal configurations that expose problems during unusual events, and operational mistakes, frequently caused by not understanding the root cause of the problem.
**Extreme Workload Spikes**

z/OS is designed to support execution of multiple, disparate workloads with very efficient use of system resources in order to meet service requirements. A key function that determines the allocation of system resources on behalf of different workloads is z/OS Workload Manager (WLM). WLM is designed to help other systems software (e.g., CICS®, IMS®, JES2, etc.) prevent unexpected workload spikes from overloading the system and causing problems that can lead to reduced system availability. The Workload Manager component of z/OS monitors the sysplex and determines the resources that should be given to each item of work to meet the service goals defined. WLM continuously evaluates metrics to recommend a new target execution environment and balances work to achieve optimal performance results. In this way, critical work continues processing and the only processing impact is to discretionary work that has a lower importance or is not time sensitive.

**Configuration Errors**

One function of z/OS that aims to prevent configuration mistakes and system stresses from causing system events is the IBM Health Checker for z/OS. IBM Health Checker for z/OS can dynamically detect when the configuration of a z/OS system does not meet recommended IBM best practices. In z/OS v2.3, there are over 250 health checks spanning most components of z/OS. These checks come with default values and importance but can be tailored to meet specific installation needs. ISVs also provide a number of health checks to suggest configuration updates on systems using their products. In sum, health checks are designed to provide an early warning system intended to help operators prevent many configuration events from causing additional problems.

**Operational Errors**

To reduce the probability of operational errors, like failure to respond to a critical message or an incorrect message response, z/OS automation can gain control for messages issued by z/OS components. In z/OS, almost every native Write to Operator (WTO) message has a unique message id whose format never changes, allowing automation products like IBM System Automation for z/OS to process messages consistently and use policy-based rules and actions to recover from problematic events. Such products are designed to provide immediate and predictable reactions to routine messages in order to help improve the resiliency of your z/OS solution.

**Masked Failures**

Masked failures, which are typically less visible, are those where z/OS detects a problem and takes corrective action, shielding the impact of the failure from the user. z/OS uses a layered approach to mask failures, starting with seamless recovery, then containment—or isolation of the failure to the smallest unit of work. The z/OS configuration is designed to have no single points of failure — a powerful approach to availability management because redundancy can help contain the impact of a failure. In addition, z/OS is designed to execute only a minimal set of recovery software necessary in order to restore health.
Examples of how z/OS can mask failures by avoiding single points of failure:

• z/OS image failure: A resilient solution requires eliminating single points of failure within the hardware configuration and the software stack. Single points of failure can be avoided by defining redundant components, such as processors, memory or servers, that deliver the application service. For instance, using two z/OS systems, two database manager instances (both being able to update the same database), two Coupling Facilities to share the data across the Sysplex, and so on, can provide extra layers of protection.

• Parallel Sysplex® is a clustered computing environment designed to provide infrastructure for sharing databases with update integrity across systems, and to enable work to be automatically routed to the most appropriate system in the event of a problem. Multiple copies of the production environment allow applications to continue to run on other systems in the event of a planned or unplanned outage, thus shielding the outage from users. Customers can also restart the impacted subsystems on another system in the sysplex, pending the recovery of the failed system. With its high levels of sharing designed throughout the stack, Parallel Sysplex is engineered to be a highly robust general-purpose compute cluster. Parallel Sysplex is engineered to identify and recover from failures “under the covers,” in order to reduce the impact to customers.

• Storage subsystem failure: z/OS provides the capability to mask a storage subsystem failure using HyperSwap®, such as: Basic HyperSwap and GDPS®, described below.

Failure and restart management processes can be initiated automatically based on policy. Furthermore, database updates can be mirrored in active-active GDPS configuration to maintain near-continuous availability for all databases accessed. (GDPS/Active-Active is a disaster recovery solution, using two or more physical sites separated by unlimited distances, which run the same workloads and data, designed to provide cross-site workload balancing and continuous availability.)

HyperSwap and Continuous Availability

HyperSwap is designed to help broaden the continuous availability attributes of z/OS by extending Parallel Sysplex redundancy to disk subsystems and enabling storage failover.

A planned HyperSwap function can:

• Transparently switch synchronous DASD replication (Metro Mirror) disk subsystems with secondary (target) Metro Mirror disk subsystems for a planned reconfiguration.

• Perform disk configuration or planned site maintenance without requiring applications to be quiesced.

In the event of unplanned outages of the primary disk subsystems, HyperSwap is intended to improve availability by helping users transparently switch to secondary Metro Mirror disk subsystems, in order to allow production systems to remain active without disruption.

Basic HyperSwap is designed to broaden z/OS continuous availability by extending Parallel Sysplex redundancy to disk subsystems. With Basic HyperSwap, disk failover operation is automated under the control of z/OS, and is designed to eliminate the need for IPLs, thus avoiding an applications outage. HyperSwap is intended to deliver high availability for a local single site Parallel Sysplex.
GDPS HyperSwap is designed to support a full enterprise-wide solution, including services to manage high availability across multiple platforms and sites. It is engineered to eliminate storage management as a single point of failure in disaster recovery plans. GDPS HyperSwap combines z/OS high availability in the same data center along with out-of-region disaster protection.

**Hard Failures**

The first type of failure that is visible to users is typically a hard failure, which can occur when a component of the software stack recognizes that it is unable to complete a requested action. An example is when a program references a virtual address that has not been allocated or requested. The z/OS response is to fail this request, often leading to abnormal termination (abend).

Therefore, one of the first principles of building a resilient IT solution is to assume that hard failures will occur. z/OS has adopted this principle and has been built assuming that failures occur. For over three decades, the z/OS operating system has included design elements to help reduce the number and impact of hard failures. These design efforts include:

- Isolating the fault when it occurs to help prevent sympathy sickness
- Capturing correct diagnostic information to help determine the root cause of the problem
- Deploying a robust quality management system to identify areas where new innovative testing approaches are needed to improve resiliency
- Minimizing the impact of a hard failure by failing the smallest amount of work and restarting the failing work as quickly as possible
- Identifying and preventing future problems using the insight derived from a review of the problem cause using a quality management system

**Soft Failures**

A soft failure occurs when unusual but permitted actions occur within a short time period and prevent the solution from delivering the desired function (hence the phrase, the system is “sick, but not dead”). Soft failures generally look different every time they occur, and their complexity makes them particularly difficult to detect, trace, and repair in order to keep the system functioning at committed service levels.

Some examples of Soft Failures:

- **Serialization:** A long running job holding an exclusive enqueue for a data set does not complete normally. A second job starts, gets an exclusive enqueue on a different dataset, and then requests the enqueue held by the first job. Now the first job requests the enqueue held by the second. This classic serialization deadlock (a deadly embrace) happens because the first job does not complete when expected.
- **Recurring failure:** A configuration or application change deployed during a change window causes a failure only when a specific, often atypical, request is made. During normal business hours, when transactions arrive, the system becomes more heavily loaded, causing request queues to gradually lengthen. Eventually, the backlog causes delays in the expected, typical transactions and these transactions time out, causing failures.
• Memory management problem: A low priority task allocates a larger area of virtual storage than usual or holds onto storage for a long period of time, which blocks high priority work from allocating the virtual storage needed. The soft failure occurs when the higher priority work fails, restarts, or slows down because it has to reduce the amount of data cached.

Addressing Soft Failures
To address soft failures and anomalies, customers can benefit from reliable, real-time, actionable alerts. The detection of soft failures in real time can be quite difficult and typically requires an understanding of past events and historical trends to better diagnose abnormal behavior.

To aid in the detection, diagnosis and mitigation of soft failures and anomalies, z/OS employs tools such as IBM Z IT Operations Analytics, Predictive Failure Analysis®, and Runtime Diagnostics, all of which are described below.

III. Tools Used to Detect Anomalies

IBM Z Operations Analytics
IBM Z Advanced Workload Analysis Reporter, part of IBM’s Z Operations Analytics, provides a solution for detecting and diagnosing anomalies in z/OS systems. This technology is designed to create a model of normal system behavior based on prior system data and use pattern recognition to identify unexpected messages in current data from the z/OS systems that it is monitoring. It builds a model of expected message patterns given a historical (recommended 90-day) baseline of your system. IBM’s analysis of events provides near real-time detection of anomalies that you can view through a graphical user interface (GUI).

With today’s high volume and complex environments, the quantity of operations log (OPERLOG) messages (can be hundreds per second) have reached a level that can no longer be consumed by a programmer. IBM Z Operations Analytics is designed to help extend availability through machine learning, pattern recognition, and statistical analysis of these high volume OPERLOG messages. It looks for unexpected patterns and anomalies in order to provide faster detection of unusual or rare messages. Such messages might be indicative of a problem and can help expedite further problem analysis. By consolidating message traffic and analyzing messages automatically, it can help pinpoint abnormalities that might be causing problems.

Detecting abnormalities quickly also allows you to investigate and solve issues that might cause similar system problems in the future if these messages had been undetected or ignored. IBM Z Operations Analytics can also be useful in identifying “sick but not dead” conditions where the system is performing sub optimally.

IBM Z Operations Analytics supports multiple analytics platforms including Splunk, Elasticstack, and the included IBM Operations Analytics Log Analysis (IOALA). IOALA runs as an independent application on Linux on IBM Z, separate from z/OS, but can consume z/OS data, allowing its use even if the z/OS operating system is non-functional.
IBM Z Operations Analytics v3.2 announcement included a statement of direction to leverage IBM Watson Machine Learning for z/OS designed to advance the existing anomaly detection capabilities.

**Predictive Detection of Soft Failures**

z/OS Predictive Failure Analysis (PFA) is designed to detect the occurrence of soft failures so an operations team can take appropriate action to mitigate the event. It is also designed to help operations personnel convert soft failures to correctable incidents by issuing alerts when it detects a problem, or when anticipating a likely problem.

PFA gathers information to model expected normal behavior of system activities and uses that information to help detect abnormal behavior. Such abnormal behavior can be indicative of a suspected soft failure. PFA is designed to reduce the amount of configuration work that would be needed to detect these events through other means and is designed to improve the accuracy of warnings. PFA works well with existing monitoring and automation products such as IBM Health Checker. For example, PFA monitors for common storage and address space virtual storage exhaustion, as well as many other “event arrival” health checks.

PFA collects data from the individual system’s components and determines what is normal, detecting resource trends that are occurring, with the goal of determining patterns. It uses advanced algorithms to determine if an abnormal condition exists or if the current trend indicates a future exhaustion. For example, PFA invokes Runtime Diagnostics to better determine whether an address space is dormant, hung, or looping.

In order to cover a wide spectrum of soft failures, PFA’s “layered approach” focuses on tracking metrics from different layers of the software stack as well as resource usage. In addition, it is designed to analyze abnormal rates of events, as they can also be indicative of different types of system problems.

**Runtime Diagnostics**

While PFA monitors the system for evidence of soft failure symptoms, z/OS Runtime Diagnostics can help the operations team analyze and diagnose a system with a potential problem or soft failure. Runtime Diagnostics is a point-in-time diagnostics feature of z/OS that examines the system for problem symptoms. It is designed to rapidly examine the system, including the sysplex, for problems using techniques similar to those a skilled system programmer might employ. These tasks include examining the sysplex message log for critical messages, identifying address spaces with high processor usage, analyzing various types of contention that include ENQ, GRS latch contention and z/OS UNIX® file system latch contention, identifying JES2 health exception events and analyzing other key contributors to soft errors.

Runtime Diagnostics is not run continuously; it is intended to be run only if you experience system degradation, or if you want to check for potential problems.
IV. Handling System Failure Conditions

Building a resilient IT solution requires addressing challenges that range from handling spikes in workload to detecting when a series of unusual events occur and cause the solution to stop working. A resilient and highly available system requires not only that the installation exploit features offered by the operating system, but also to plan for the handling of problems identified by z/OS Health Checker and other error detection mechanisms.

Actions to Take If a Failure Occurs

As mentioned, one of the first principles of building a resilient IT solution is to assume that failures will occur. Just as z/OS is designed to perform critical actions whenever a failure is detected, similar techniques are recommended when building your own IT solution:

- Deal with the failure
- Re-drive the work or redirect the work to a peer system, subsystem or application instance in the sysplex
- Clean up after the failure to prevent a recurrence
- Isolate the fault to the smallest possible unit of work to prevent a small problem from escalating
- Capture pertinent diagnostic information to help determine the cause of the problem

If a system failure is suspected, z/OS provides extensive capabilities to help the operations team confirm that the event is really a soft failure and then diagnose the cause.

- A system monitor like OMEGAMON® XE can be used to examine the system for problem clues based on resource utilization, based on information gathered by system management facility (SMF). System monitors provide useful problem clues by helping with contention analysis of system and application enqueues and latches.
- Other z/OS components provide built-in detection of soft failures. For instance, JES2 monitors its work queues, and the catalog component monitors catalog tasks.
- XCF and XES provide a detection mechanism for soft failures involving systems interacting in the sysplex with XCF heartbeats and partner system status updates recorded in the Couple Data Set.
- Any sysplex-wide program that becomes impaired, resulting in communication stalls, marks the offending system as needing removal from the sysplex. This helps isolate and contain the problem and prevents the spread of “sympathy sickness”.
- In addition, z/OS platform code is developed to meet internal IBM reliability, availability and serviceability (RAS) guidelines that specify diagnostic and recovery techniques to be followed in mainline programs (before a problem occurs), expectations of abend recovery (at the time of error), and data that must be captured to help facilitate diagnostics (after the error).

The following sections demonstrate the types of error handling that enables z/OS to deliver highly reliable functions. Note than in addition to IBM solutions, IT solution owners may wish to choose hardware and software components that are extremely reliable, able to detect issues as close to the source as possible, ensure processes in error are terminated with minimal impact and ensure that appropriate diagnostic information is captured.
• **Recovery Processing**
  If a failure is reported to the operating system, z/OS component Recovery Termination Manager leverages recovery routines that the mainline processing has established to tailor recovery handling for each program currently executing, starting with the one that encountered the failure. These recovery routines capture diagnostic data and help determine how to recover – to continue processing where it left off (or “retry”), or to terminate the program’s execution and surface the error.

  Recovery routines will automatically attempt to recover a failing program. If a retry operation is not possible, the recovery routine will clean up resources associated with the failure, isolating the failure to that task. There’s a delicate balance between allowing a program to continue processing vs. terminating that program. The choice of an appropriate recovery action can impact the likelihood that errors may continue. Program termination provides a clean base to allow the system to re-attempt the process. A component’s recovery must also free unused resources to ensure that inadvertent serialization dependencies are avoided.

• **Fault Isolation**
  Similar to the support provided by z/OS for error recovery, the hardware and operating system offer support to isolate faults to the smallest units of work. In a Parallel Sysplex, it is critical to rapidly remove a failing part of the system or the entire system from the cluster before it can impact the cluster more broadly, causing “sympathy sickness”. Sysplex Failure Management (SFM) is a z/OS function that is designed to detect when either a system or a member is not working; it then notifies operations or removes the failing system from the cluster.

  The ability to detect an unresponsive z/OS image and remove it from the sysplex is critical to isolating a cluster-sensitive fault without causing additional problems. Because the z/OS image may no longer be working, that image may be unable to provide the sysplex with timely status updates. Hardware status detection provided by z/OS is designed to help speed detection of a problematic image and reset a successfully removed z/OS image.

• **Finding the Root Cause of a Problem**
  In some cases, customers are unable to determine or recreate the exact sequence of events necessary to reproduce a failure. To capture the correct data, z/OS provides memory dumps to internal traces. Based on the details, the recovery routines capture the diagnostic information, including the appropriate trace buffers and memory associated with the failing address space, and ensures that those areas are included in diagnostics. These diagnostics can aid in more rapid analysis of the problematic situation.

• **Improving Performance of the Process of Diagnostics Collection**
  If a failure occurs, a standalone dump or SVC dump may be used to help diagnose the failure. The generation of dumps can require significant processing time. In order to sustain high levels of availability and performance during dumps, flash memory can be used to improve paging performance. This is designed so that paging performance during SVC or standalone dumps, as well as during other transitional times, is dramatically accelerated. Such comprehensive diagnostic collection could otherwise cause significant delays to performance impacting the availability of critical online workloads.

  In summary, z/OS is designed to assume that unexpected masked or hard failures will occur and is prepared to collect information to debug the problem, clean up shared resources, and isolate the problem, as well as addressing possible side effects.
V. Disaster Recovery
Up to now, our discussion has focused on deploying and managing a system, sysplex or enterprise with very strong high availability or continuous availability characteristics. This includes mitigating the impact of planned or unplanned outages within a data center and planning for data availability. Disaster recovery is focused on recovering from events that caused a full data center outage.

It is Critical to Separate Planning for High Availability vs. Disaster Recovery.
Recovery objectives for component (hardware or software) failures experienced within a data center are often expressed in seconds or minutes, depending on technology used and how well the installation’s system recovery actions are automated. Recovery objectives for disasters could be as high as hours, including the time it takes to switch operations over to an alternate data center, whether it’s a “hot” environment that just needs important software to be re-initialized, or a “cold” environment that requires hardware to be powered up, operating systems restarted, databases recovered and production workloads up and running, using replicated data.

Many companies use GDPS to replicate production data to an alternate data center(s), with all recovery tasks fully-automated to occur at “machine speed,” which are designed for very fast recovery and minimal, if any, data loss. The recovery is typically expressed in terms of RTO and RPO.

Recovery Time Objective (RTO) describes the expected time to recover from a failure when service is restored, to support the business user. RTO is often expressed in terms of an organization’s tolerance for loss of business from a workload or system outage.

Recovery Point Objective (RPO) describes the amount of data loss the business can tolerate based on the time it takes to restore the data from a backup source. Typically, RPO is supported by IBM storage technology, capable of recovering data with little or no loss.

Most large businesses have investments in immediate DASD data replication to enable system recovery with zero data loss. IBM’s data replication technologies include the following, and are often used in combination, depending on the organization’s recovery requirements:

• Synchronous replication technology on the DS8000®, Metro Mirror, is based on Peer-to-Peer Remote Copy (PPRC), where every I/O is synchronously copied to a secondary DASD volume hosted on another system, intended for zero data loss. The DASD pairs need to be within 200 fiber km in a GDPS Metro configuration (formerly known as GDPS/PPRC), and 303 fiber km from a PPRC perspective. Synchronous data replication is typically used to replicate data to alternative systems within the data center.

• Asynchronous replication technologies in the DS8000, Global Mirror and z/OS Global Mirror, are in place to handle data replication between sites at longer distances (as compared to Metro Mirror). Global Mirror is based on Extended Remote Copy (XRC), where every I/O is queued to a secondary system over a longer distance, designed for minimal data loss.

• z/OS HyperSwap replication is a special Metro Mirror replication method for storage systems that are connected to an IBM z/OS system. HyperSwap automates the failover of I/O from the primary logical devices to the secondary logical devices in the event of a primary disk storage system failure.
**GDPS Delivers Fully-Automated, Multi-Site Disaster Recovery**

GDPS is designed to provide not only resource sharing, workload balancing, and near continuous availability benefits of a Parallel Sysplex environment, but it can enhance the capability of an enterprise to recover from disasters and other failures and to manage planned exception conditions. GDPS can allow a business to achieve its own continuous availability and disaster recovery goals. With IBM GDPS, you can be confident that your key business applications will be available when your employees, partners and customers need them.

**GDPS solutions include:**

- **GDPS HM:** GDPS Metro HyperSwap Manager (GDPS HM) provides an entry level multisite disaster recovery solution at a cost-effective price. HyperSwap provides the ability to dynamically switch to secondary volumes without requiring applications to be quiesced. Typically done in 3–15 seconds in actual customer experience, this provides near-continuous data availability for planned actions and unplanned events.

- **GDPS Metro:** GDPS Metro is designed to be a near continuous availability and disaster recovery solution. GDPS Metro has all the function as GDPS HM and is also designed to fully automate the recovery at the remote site.

- **GDPS XRC:** GDPS Global – XRC (GDPS XRC) is a highly scalable asynchronous remote copy solution for z/OS and Linux on IBM Z data. Based upon z/OS Global Mirror, it is a combined hardware and software asynchronous remote copy solution. Since z/OS Global Mirror uses asynchronous data replication, the secondary site can be thousands of miles from the primary site.

- **GDPS GM:** GDPS Global – GM (GDPS GM) is an asynchronous remote copy solution for z/OS and non-z/OS data. Based on the DS8000 Global Mirror, it is a hardware based asynchronous mirroring solution that is designed to maintain a consistent copy of data at virtually unlimited distances and is intended for minimal impact to application response time.

- **Benefits of Combining the Products Above (GDPS MGM & GDPS MzGM):** Three Site or Four Site Solutions are available for businesses requiring the benefits of both synchronous and asynchronous remote copy. Synchronous remote copy using GDPS Metro and GDPS HM provides benefits such as near-continuous availability using HyperSwap and the ability to configure for zero data loss. Asynchronous remote copy using GDPS XRC or GDPS GM is designed to provide benefits such as protection from regional disasters with little to no application impact. To provide for this requirement, GDPS supports three-site configurations to help provide maximum availability across the widest range of possible scenarios.

- **GDPS Continuous Availability:** GDPS Continuous Availability (GDPS AA), previously called GDPS/Active-Active, is an asynchronous remote copy solution for select z/OS workloads. GDPS Continuous Availability is a software based asynchronous mirroring solution designed to maintain a consistent copy of data within IMS, within Db2, and within VSAM data. GDPS AA is designed to support separate sysplexes run in sites separated by extended distances. It is designed for recovery time measured in seconds, and for minimal impact to application response time.
VI. Cyber-Resiliency and Data Corruption Detection

Cyber-Resiliency is often an important part of an organization’s strategy that unites information security, network and business continuity into a comprehensive resiliency strategy to detect unknown threats, respond to cyber outbreaks, recover access to critical data and protect against future attacks. Many organizations focus on data protection as part of their planning, including how data is stored and accessed, as well as compliance and audit requirements. Sensitive data needs to be properly protected on the cloud and shared storage infrastructure.

But what if there’s a catastrophic event where the production environment is found to be unusable and repair or partial restore of the data is not viable? This could be due to unintended deletion, overwrite or encryption of data, or other large-scale events that logically destroy all continually replicated copies of the data.

Logical Corruption Protection

Logical Corruption Protection is designed to help clients recover from malicious attacks on their data, or unintended corruption, by preserving up to 10 periodic point-in-time copies prior to the corruption event. The backup copies are written using the DS8000® Safeguarded Copy feature. It creates safeguarded backups that are designed to be inaccessible by the host system and protects these backups from corruption that can occur in the production environment. Safeguarded copies can be created on a regular basis, such as hourly or daily. To restore, FlashCopy® is used to copy data back from a protected copy after identifying the desired recovery point.

VII. Engineering Resiliency into the Solution

Testing approaches are also used to help improve the delivery of resilient systems.

IBM design principles assume that testing emulates the behavior of customer systems. The ability to remove problems that are likely to be experienced by the customer helps improve the resiliency of the solution. To test a system, IBM creates environments that closely resemble customers' environments. Effective test case coverage helps provide assurance that the test cases are sufficient to represent realistic customer scenarios.

Workload profiling is a proactive, comprehensive characterization process used to verify and improve workload drivers by adopting test characteristics that are similar to customer production environments and applications. By stress testing z/OS in this manner, IBM’s z/OS test organization seeks to improve resiliency by helping with early discovery of defects that customers might otherwise encounter on their own.

Validating a Resilient Solution

The experience of building z/OS, as a critical part of a resilient solution, requires a comprehensive, robust, and aggressive quality management system to identify where the process and the delivery of the solution need improvement. As part of this quality management process, IBM’s z/OS test organization validates that testing covers different customer environments and reflects important cross-image problems that may be incurred by customers.
VIII. Conclusion

As IT solutions support more critical function and continue to drive significant business impact, the need to provide resilient IT solutions increases. Solutions must enable systems to survive failures and recover quickly. The recommended keys to building resilient IT solutions are:

• Configure your company’s infrastructure to avoid single points of failure (SPOF) on every hardware and software component:
  – CECs (processors)
  – Storage devices
  – Logical operating system images
  – Subsystem images
  – Application images
  – Communication devices
  – Access paths

• Address the full range of challenges that your solutions will face
• Build your IT solutions with an attention to detail
• Use multiple continuous feedback loops to validate that components within the solution, and the solution itself, are resilient
• Continuously strive to deliver better, more available, large system solutions
• Use intelligent software to identify potential problems before further outages occur
• Include cyber resiliency and data resiliency in your planning
• Use solutions to isolate failures

Building a resilient solution is a journey, not a destination, so it requires continuous improvement throughout the entire design, development and testing process. But if the system is designed to survive failure, the effort is well worth it.
References

- “HyperSwap Helps Keep Storage From Becoming a Single Point of Failure”: http://ibmsystemsmag.com/aix/administrator/highavailability/hyperswap_ha/
- “Introduction to Flash Express”: https://www.ibm.com/downloads/cas/4AMDK9BX
- “z/OS Parallel Sysplex Operational Scenarios”: http://www.redbooks.ibm.com/abstracts/sg242079.html