IT Challenge – Building Systems to Survive Failures
Table of Contents

Introduction .................................................................................................................................3
  Introduction: The Critical “IT NEWS EVENT” .........................................................................3
Resilient IT Solutions and Understanding Failures .....................................................................4
  Avoidable Failures ..................................................................................................................4
  Masked Failures ......................................................................................................................5
  HyperSwap and continuous availability ..................................................................................6
  Hard Failures ..........................................................................................................................7
  Soft Failures ............................................................................................................................7
Tools Used to Detect Anomalies .................................................................................................8
  Handling System Failure Conditions ......................................................................................9
Engineering Resiliency into the Solution ..................................................................................11
Conclusions ..............................................................................................................................12
References ................................................................................................................................13
Introduction
Over the past decade IT systems have become more reliable. However, when IT failures or large problems do inevitably occur, the collateral damage has increased. As IT systems become pervasive, there is a need to move beyond highly available systems to highly resilient systems; that is, systems able to survive failures that affect a business or enterprise. A resilient system can isolate, remove, tolerate and recover from component failures so that application and business impact are minimized. Based on our 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 z/OS® 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 IBM’s experience and discusses:

- Resilient IT Solutions- Understanding Different Types of Failures
- Tools used to Address Anomalous Situations
- Handling Failure Conditions
- Engineering Resiliency into The System

Introduction: 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, with the usual result of angry customers, upset stakeholders and brand erosion. With IT providing critical solutions for customers around the world, IT solutions need to make every effort to avoid an external event of such proportions.

Resilient systems should be able to detect and recover from system issues thus avoiding such “IT News Events” wherever possible. To most of these stakeholders it doesn’t matter what causes an event, only that the event occurred and is resolved. To minimize the risk of an external IT News event customers need a resilient solution, a solution that “has the ability to absorb failure(s) or contain damage while continuing to function.” The assumption is that events will occur, and that the system must be able to handle them and recover from them seamlessly.

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

A review of publicly available and critical problems reported on z/OS indicates that most enterprise-class IT solutions actually require a combination of events to cause a problem that is likely to generate an external IT news event. A resilient IT solution needs to contain a combination of unexpected events when they occur over a short period of time. It needs to recover from that combination of events, to isolate the problem 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 such resiliency into its solutions, and can serve as an example of what customers can do as well to fortify their IT solutions. We discuss:

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

Types of Failures

A key to building a resilient solution is to understand the types of unexpected problems or challenges IT faces. IT solutions can be categorized into four categories:

- **Avoidable failures** that could have been prevented using the capability of solution components like an operating system such as z/OS.
- **Masked failures** where a component -even application code- detects a problem, corrects or mitigates the problem, and successfully hides the problem from users.
- **Hard failures** where a solution component including the application, detects a problem and fails immediately or when the component’s recovery is unable to successfully complete and fails.
- **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 are particularly difficult to diagnose.

Once a failure occurs it can cause a cascade of problem that increases the impact of the original event, often described as sympathy sickness. In these situations, the observed failure is merely a reaction to one or more failures of different types that occur in related areas of processing.

**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 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. A key function that manages system resources on behalf of different workloads is z/OS Workload Manager (WLM). WLM dynamically reallocates system resources between applications to prevent unexpected workload spikes from overloading the system and causing problems that can lead to reduced system availability. When an extreme workload spike occurs, WLM can automatically defer discretionary work until the spike has passed, allowing critical work to continue. 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 V1R13 there are over 185 health checks spanning most components of z/OS. These checks come with default values and importance, but they 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. Health checks provide an early warning system to help operators prevent certain 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, or WTO message, has a unique message id whose format never changes allowing automation products like Tivoli® System Automation to process messages consistently and use policy-based rules and actions to recover from problematic events. Such products provide immediate and predictable reactions to routine messages, helping to improve the resiliency of your z/OS solution.

Masked Failures
A less visible failure is a masked failure where z/OS functions detect 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 providing seamless recovery, then containment—or isolation of the failure to the smallest unit of work. As a result the z/OS configuration can be one with no single points of failure — a powerful approach to availability management because it contains the failure as close to the source as possible. It also has the benefit of requiring successful execution of only a minimal set of recovery software 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 that deliver the application service—that is, at least 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.

- Parallel Sysplex® is a clustered computing environment that provides infrastructure to share databases with update integrity across systems and enables work to be automatically routed to the most appropriate system in the event of a problem. Multiple copies of production environment allow applications to continue to run on other systems in the event of a planned or unplanned outage shielding the outage from users. Also, you can restart the impacted subsystems on another system in the sysplex, pending the recovery of the failed system. With its high levels of sharing throughout the stack, Parallel Sysplex is often considered world's most robust general purpose compute cluster. Parallel Sysplex is designed to identify and recover from failures “under the covers” reducing the impact to customers.

- Failure and restart management processes can be initiated automatically based on policy. Furthermore, database updates can be mirrored in active-active configurations to maintain near-continuous availability for all database accessed.

- Storage subsystem failure: z/OS provides the capability to mask a storage subsystem failure using HyperSwap™. There are two forms of HyperSwap: Basic HyperSwap and Geographically Dispersed Parallel Sysplex™ (GDPS®). Basic HyperSwap is a single-site high availability solution delivered as part of z/OS and managed by TotalStorage® Productivity Center for Replication (TPC-R).

HyperSwap and continuous availability

HyperSwap helps broaden the continuous availability attributes of z/OS by extending Parallel Sysplex redundancy to disk subsystems. 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 applications to be quiesced.

Unplanned HyperSwap improves availability by helping users transparently switch to secondary Metro Mirror disk subsystems in the event of unplanned outages of the primary disk subsystems, allowing production systems to remain active.

Basic HyperSwap broadens z/OS continuous availability by extending Parallel Sysplex redundancy to disk subsystems. With Basic HyperSwap, the entire disk failover operation is managed in seconds, fully automated, and controlled through z/OS, eliminating re-IPLs, and thus avoiding an applications outage. HyperSwap is intended to deliver high availability for a local single site Parallel Sysplex.

On the other hand, GDPS HyperSwap provides a full enterprise wide solution including services to manage high availability across multiple platforms and sites, and fully eliminates 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. A hard failure occurs when a component of the software stack recognizes that it is unable to complete the requested action. A classic example of a *hard failure* is when a program references a virtual address that has not been allocated or requested. The response on z/OS is to fail this request because the value at the specified virtual address is undefined.

*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 z/OS and its predecessors have been working to reduce the number and impact of hard failures. These efforts include:

- Isolating the fault when it occurs to prevent sympathy sickness.
- Capturing correct diagnostic information to 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.
- Finding and preventing future problems using the insight derived from a review of the problem cause using a robust quality management system.

**Soft Failures**

A soft failure occurs when unusual but permitted actions occur within a short time period preventing 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 and trace.

### 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 more requests occur and the system is heavily loaded causing request queues to gradually lengthen. Eventually, the backlog causes delays in typical transactions so that the typical 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 need reliable, real-time, actionable alerts. The detection of soft failures in real time can be quite difficult and requires either thresholds or statistically based algorithms capable of predicting normal behavior. A statistically based algorithm to detect abnormal behavior is an attractive option when threshold values cannot be defined by the customer or by IBM. To identify candidate components responsible for abnormal behavior, z/OS makes use of both thresholds and statistical algorithms, some of which are described below.

Tools Used to Detect Anomalies
IBM zAware
The newly available IBM System z® Advanced Workload Analysis Reporter (IBM zAware) provides a smart solution for detecting and diagnosing anomalies in z/OS systems on the IBM zEnterprise® EC12 (zEC12) server. IBM zAware creates a model of normal system behavior based on prior system data, and uses pattern recognition to identify unexpected messages in current data from the z/OS systems that it is monitoring. It builds a mathematical model of expected message patterns given a historical (recommended 90 day) baseline of your system. IBM zAware’s analysis of events provides near real-time detection of anomalies that you can easily view through a graphical user interface (GUI).

IBM zAware extends availability through machine learning, pattern recognition, and statistical analysis of OPERLOG messages. IBM zAware looks for unexpected patterns and anomalies to provide fast detection of unusual or rare messages. Such messages might be indicative of a problem and can help expedite further problem analysis. Why is this task difficult to accomplish through other means? The quantity of OPERLOG messages today is no longer consumable by the system programmer (these can range in the hundreds of messages per second). IBM zAware can consolidate message traffic on your systems and analyze messages automatically to 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 zAware is also useful for “sick but not dead” conditions where the system is performing sub-optimally.

IBM zAware applies domain knowledge of z/OS and knows which messages are critical so that if they do occur, a system problem is likely. IBM zAware also eliminates messages that could skew the analysis and that are considered “noise” on the system. Therefore, there is intelligence to determining which messages might warrant further attention. Without IBM zAware, operator attention to the incorrect cause of the problem might lead to unproductive fixes, ineffective changes, and ultimately missed service levels.

Even though IBM zAware has z/OS domain knowledge, it runs as an independent pre-integrated firmware stack in its own partition separate from z/OS. This reduces the resources required of the operating system and allows IBM zAware to be used even when the operating system is non-functional.
Predictive Detection of Soft Failures

z/OS Predictive Failure Analysis® (PFA) detects the occurrence of a set of soft failures so an operations team can take appropriate action to mitigate the event. It converts 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 determine when abnormal behavior is occurring. Such abnormal behavior is an indication of a suspected soft failure. As a result, it reduces the amount of configuration work that would be needed to detect these events through other means, and improves the accuracy of the warning. Using IBM Health Checker for z/OS, PFA works well with existing monitoring and automation products.

PFA collects data from the individual system’s components and determines what is normal, detecting resource trends that are occurring. It uses advanced algorithms to determine if an abnormal condition exists or if the current trend indicates a future exhaustion.

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 ranging as well as resource usage. Other analysis is focused on “event rate analysis” as abnormal rates of events can be indicative of different types of system problems.

Handling System Failure Conditions

Building a resilient IT solution requires addressing challenges ranging from handling spikes in work load to detecting when a series of unusual events occur that causes 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 plan for the handling of problems identified by z/OS Health Checker and other error detection mechanisms.

Actions to Take When 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 required when building your own IT solution:

- Deal with the failure
  - Re-drive the work to mask the failure, 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 allow the operations team to confirm that the event is really a soft failure and 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 through z/OS’s contention analysis for system and application use of 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, and 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-scope 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 written to meet internal IBM reliability, availability and serviceability (RAS) guidelines that specify diagnostic and recovery techniques to be followed in mainline programs (before the problem occurs), expectations of abend recovery (at the time of error), and data that must be captured to facilitate diagnostics.

The following sections demonstrate the types of error handling that enables z/OS to deliver highly reliable functions. Note: Similar to IBM, IT solution owners, too, have a responsibility 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 ensuring 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 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. In any event, a component’s recovery must free unused resources to ensure that inadvertent serialization dependencies are avoided.

- **Fault Isolation**
  Just like the support provided by z/OS for error recovery, the hardware and operating system offers 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 impacts the cluster more broadly, causing “Sympathy Sickness”. Sysplex Failure Management (SFM) is a z/OS function which detects when either a system or a member is not working and 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. Hardware status detection provided by BCPii component of z/OS helps speed up 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 a rich set of primitives from 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. To prevent the system from being overwhelmed if the same error recurs, z/OS can automatically suppress duplicate dumps.

Improving Performance of Diagnostics Collection
After a failure occurs a standalone dump or SVC dump may be used to help diagnose the failure. In order to sustain high levels of availability and performance during dumps, Flash Express, flash memory on server PCIe cards introduced with the zEC12, can be used to improve paging performance. With Flash Express, paging performance during SVC or Standalone dumps as well as during other transitional times is dramatically accelerated. Such diagnostic collection could otherwise cause significant delays to performance impacting availability of critical online workloads. The improved paging performance resulting from Flash Express cards (1.6 TB per card pair) helps shave paging delays and reduces the time that other critical systems may be impacted.

In sum, 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. Such operating system services enables you to focus on building your IT business solution, knowing that z/OS is in full control of handling error recovery.

Engineering Resiliency into the Solution
Testing approaches are also used to help improve the delivery of resilient systems. Customer profiling is used to improve the resiliency of z/OS by creating a test environment that models the critical operating system behaviors and usage of actual z/OS customers.

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 the customers' environments. Effective test cases coverage helps provide assurance that the test cases are sufficient to represent realistic customer scenarios.

“Workload profiling” is a proactive, comprehensive characterization process that can be used to verify and improve workload drivers by adopting test characteristics that are similar to customer production characteristics, including characteristics of customer environments and applications. The process employs an in-depth examination of the customer environment, applications, and defect data. Understanding and implementing production workload characteristics within IBM’s z/OS test organization can help 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. On z/OS this includes validating that testing matches different customer environments; it also involves having customers identify potentially important cross-image problems. Building a resilient solution is a journey not a destination, so it requires continuous improvement throughout the entire development and testing process.

Conclusions
As IT solutions drive more critical function and continue to have significant business impact, the need to provide resilient IT solutions increases. Solutions must enable systems to survive failures and recover. The 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:
  - From CECs to storage devices,
  - To logical operating system images,
  - To subsystem images,
  - To application images,
  - To communication devices,
  - To access paths,
  - Avoid SPOFs wherever possible
- 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 IBM solutions to isolate failures so you can be proactive in restoring service

Building a resilient solution is a journey which requires many parts and many disciplined steps, but the results are well worth the effort, as demonstrated by the resilient function delivered in z/OS.
References

- HyperSwap, “z/OS Basic HyperSwap and GDPS HyperSwap Overview Presentation”:
  www.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/PRS3168

- “IBM GDPS active/active continuous availability”:

- “IBM zAware”:

- IBM z/OS Parallel Sysplex Operational Scenarios:
  http://www.redbooks.ibm.com/abstracts/sg242079.html

- “Introduction to Flash Express”:

- Predictive Failure Analysis: “z/OS Problem Management”:

- “z/OS Parallel Sysplex Operational Scenarios”:
  http://www.redbooks.ibm.com/abstracts/sg242079.html