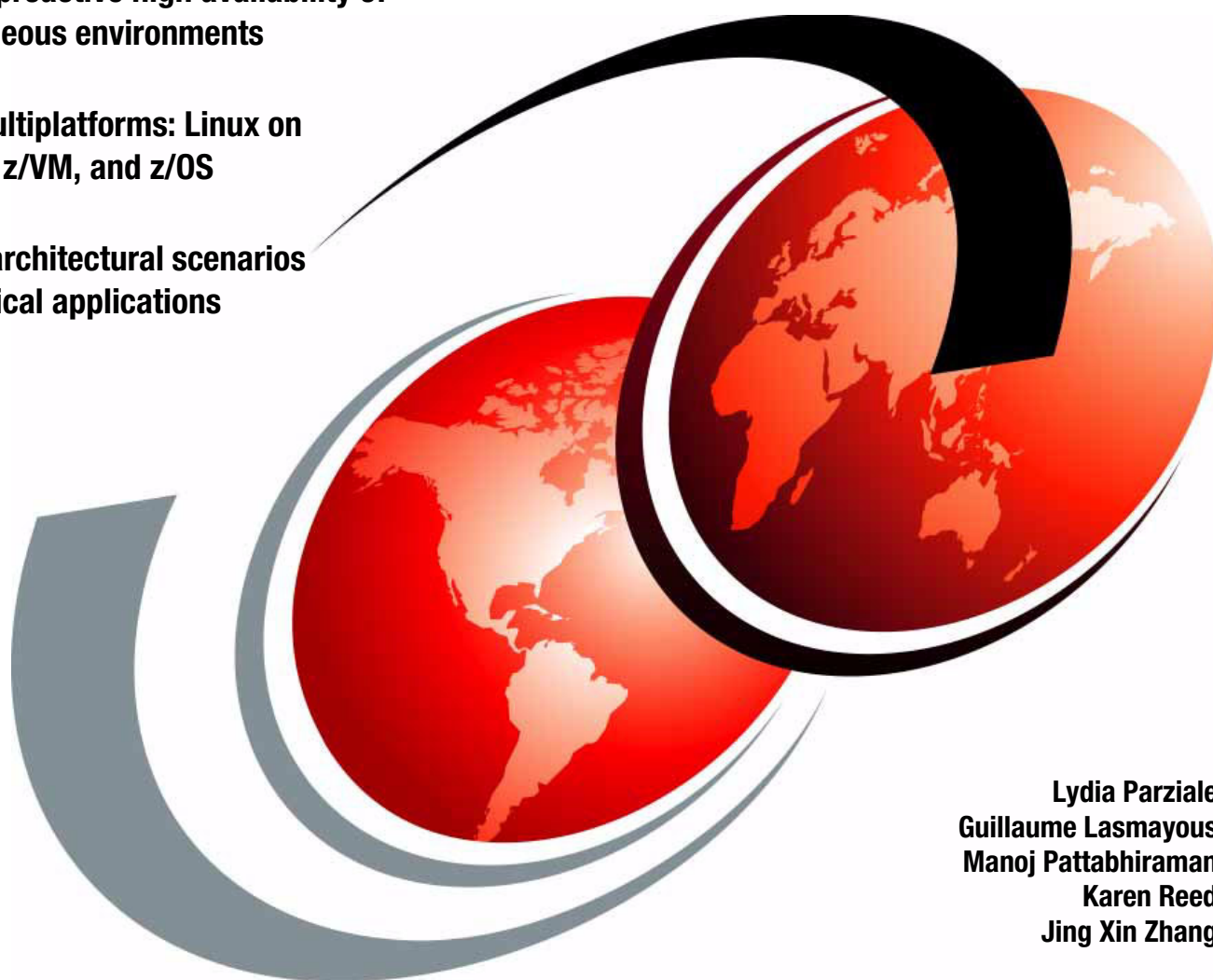


End-to-End High Availability Solution for System z from a Linux Perspective

Achieves proactive high availability of
heterogeneous environments

Covers multiplatforms: Linux on
System z, z/VM, and z/OS

Includes architectural scenarios
and practical applications



Lydia Parziale
Guillaume Lasmayous
Manoj Pattabhiraman
Karen Reed
Jing Xin Zhang

Redbooks



International Technical Support Organization

**End-to-End High Availability Solution for System z
from a Linux Perspective**

October 2014

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (October 2014)

This edition applies to IBM Tivoli System Automation for Multiplatforms v4.1, DB2 v10.5, SUSE Linux Enterprise Server 11 SP 3, Red Hat Enterprise Linux v6.4, and IBM z/VM 6.3.

© Copyright International Business Machines Corporation 2014. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	vii
Trademarks	viii
Preface	ix
Authors	ix
Now you can become a published author, too!	x
Comments welcome	x
Stay connected to IBM Redbooks	xi
Chapter 1. High availability fundamentals	1
1.1 High availability concepts	2
1.2 High availability terminology	3
1.2.1 Availability processes and targets	3
1.2.2 Outage and recovery terminology	4
1.2.3 HA architecture terminology	6
Chapter 2. Business and IT availability	11
2.1 Business and IT resilience	12
2.1.1 Regulations	14
2.1.2 Business requirements	14
2.1.3 Other considerations	14
2.2 High availability inhibitors in IT practices	17
2.2.1 Phase 1: Planning for high availability	18
2.2.2 Phase 2: Development	18
2.2.3 Phase 3: Validation	18
2.2.4 Phase 4: Implementation and updating	19
2.3 Business impact of outages	20
2.3.1 Quantifying the cost of making services continuously available	20
2.3.2 Continuous availability versus high availability versus disaster recovery	21
2.3.3 Heterogeneous end-to-end view	22
2.3.4 Poor communication between business and IT	22
2.4 Determining single points of failure	23
2.5 High availability options	23
2.5.1 Active / active configuration	24
2.5.2 Active / passive configuration	25
2.5.3 Being realistic about the cost of continuous availability	26
2.6 Deciding on a high availability solution	26
Chapter 3. Components of high availability	29
3.1 Components of a high availability solution	30
3.2 Hardware high availability	30
3.2.1 System z hardware	31
3.2.2 IBM DS8000 series	32
3.3 Storage high availability	34
3.3.1 Storage copy services	34
3.3.2 Site replication options	35
3.4 Operating system high availability	39
3.4.1 Operating system storage high availability features	39
3.4.2 z/VM high availability features	41

3.4.3 Linux high availability features	43
3.5 Software solutions for high availability	46
3.5.1 Linux with z/OS high availability solutions.	51
3.6 Network high availability	52
3.6.1 External network infrastructure	53
3.6.2 Open Systems Adapter cards	53
3.6.3 z/VM Virtual Switch	61
3.6.4 HiperSockets	66
Chapter 4. Architectural scenarios	69
4.1 Database high availability: Two sites	71
4.1.1 High availability clustering and automation tools	73
4.1.2 Database layer	75
4.2 z/OS database scenario	76
4.2.1 Parallel Sysplex technology	76
4.2.2 DB2 sysplex	78
4.2.3 GDPS Metro and z/OS Global Mirror (z/OS Metro/Global Mirror).	80
4.2.4 DB2 Q-replication for HA.	81
4.3 z/OS and Linux scenarios	82
Chapter 5. Practical applications.	85
5.1 DB2 High Availability Disaster Recovery (HADR).	86
5.1.1 DB2 HADR setup	86
5.1.2 DB2 HADR prerequisites	86
5.1.3 Setting up the DB2 registry profile	87
5.1.4 Linux ports configuration.	88
5.1.5 Database archive setup	88
5.1.6 Backup of the primary database	89
5.1.7 Setting up the DB2 database configuration	89
5.1.8 Starting HADR	91
5.1.9 DB2 HADR takeover verification.	92
5.1.10 Why we used Tivoli System Automation.	94
5.2 IBM Tivoli System Automation for Multiplatforms	94
5.2.1 Prerequisites	95
5.2.2 Tivoli System Automation manual installation.	95
5.2.3 Preparing the node for communication	96
5.2.4 SA MP Cluster configuration.	97
5.2.5 SA MP DB2 automated failover scenarios	103
Chapter 6. Summary	115
6.1 The need for high availability	116
6.2 Applications that need to be highly available	116
6.3 High availability options available on System z.	116
Appendix A. Common TSA commands.	119
Common Tivoli system automation commands	120
Appendix B. Hints and tips during cluster setup	121
Domain nodes not communicating.	122
Node names.	122
Problems with db2haicu definition	122
Unhealthy TSA cluster state.	122
Related publications	127

IBM Redbooks publications 127

Other publications 127

Help from IBM 127

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.


COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	HyperSwap®	Tivoli®
DB2®	IBM®	z/OS®
DS8000®	Parallel Sysplex®	z/VM®
FlashCopy®	Redbooks®	zEnterprise®
GDPS®	Redbooks (logo)  ®	
Geographically Dispersed Parallel Sysplex™	System Storage®	
	System z®	

The following terms are trademarks of other companies:

Intel, Itanium, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

As Linux on IBM System z® becomes more prevalent and mainstream in the industry, the need for it to deliver higher levels of availability is increasing.

This IBM® Redbooks® publication starts with an explanation of high availability (HA) fundamentals such as HA concepts and terminology. It continues with a discussion of why a business needs to consider an HA solution and then explains how to determine your business single points of failure.

We outline the components of a high availability solution and describe these components. Then we provide some architectural scenarios and demonstrate how to plan and decide on an implementation of an end-to-end HA solution, from Linux on System z database scenarios to IBM z/OS®, and include storage, network, IBM z/VM®, Linux, and middleware.

This implementation includes the IBM Tivoli® System Automation for Multiplatforms (TSA MP), which monitors and automates applications distributed across Linux, IBM AIX®, and z/OS operating systems, as well as an IBM Geographically Dispersed Parallel Sysplex (IBM GDPS®) based solution. It includes the planning for an end-to-end scenario, considering Linux on System z, z/VM, and z/OS operating environments, and the middleware used. The TSA MP implements HA for infrastructure, network, operating systems, and applications across multiple platforms, and is compared to a Linux HA implementation based on open source Linux-HA, which is Linux only.

Authors

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York, with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include e-business development and database management technologies. Lydia is a Certified PMP and an IBM Certified IT Specialist with an MBA in Technology Management and has been employed by IBM for over 25 years in various technology areas.

Guillaume Lasmayous is an IBM Certified IT Specialist in the IBM Montpellier Client Center in Montpellier, France. He has 9 years experience with z/VM and Linux on System z, providing presales technical support to European customers considering solutions running on Linux on System z. He holds a degree in mechanical engineering from ECAM Lyon, France. A long-time Linux user and open source developer, his areas of expertise include virtualization and consolidation with z/VM and Linux on System z as well as Cloud Computing and infrastructure management solutions.

Manoj S Pattabhiraman is an IBM Certified Senior IT Specialist from the IBM STG Benchmarking Center, Singapore. He has more than 13 years of experience in System z Virtualization, Cloud, and Linux on System z. Manoj holds a Masters degree in computer applications from VIT, India. In his current role, he leads the System z benchmarking team in Singapore and also provides consultation and implementation services for various Linux on System z customers across Growth Markets. Manoj has contributed to several z/VM and Linux on System z related IBM Redbooks publications, and has been a frequent presenter at various technical conferences and workshops on z/VM and Linux on System z.

Karen Reed is a Certified IT Systems Engineering professional with experience in a broad range of hardware and software architectures. She has 20 years of experience in technical sales and implementation support of IBM system monitoring, analytics, and automation software on System z, supporting clients in the western USA. Her areas of expertise include analyzing client business needs and designing IT solutions, and project management. She has presented IBM software solutions for systems management and automation at SHARE, CMG, and IBM Cloud conferences.

Jing Xin Zhang is an Advisory System Service Representative from IBM China. He has 6 years of experience in supporting mainframe in major national banks. He also has worked as an ELS delivery team core member for TSS ELS project since 2010.

Thanks to the following people for their contributions to this project:

Rich Conway, Robert Haimowitz
International Technical Support Organization, Poughkeepsie Center

Sylvain Carta
IBM France

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:
ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

ibm.com/redbooks

- Send your comments in an email to:

redbooks@us.ibm.com

- Mail your comments to:

IBM Corporation, International Technical Support Organization
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find us on Facebook:
<http://www.facebook.com/IBMRedbooks>
- ▶ Follow us on Twitter:
<http://twitter.com/ibmredbooks>
- ▶ Look for us on LinkedIn:
<http://www.linkedin.com/groups?home=&gid=2130806>
- ▶ Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:
<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>
- ▶ Stay current on recent Redbooks publications with RSS Feeds:
<http://www.redbooks.ibm.com/rss.html>



High availability fundamentals

This IBM Redbooks publication provides an overview of potential solutions for improving availability of end-to-end workloads on System z. We discuss the decisions that go into selecting the infrastructure required to support business requirements for IT resilience. We describe the high availability environment created and tested in our System z lab. In general, we introduce the concepts of high availability, describe options for improving the availability of an IT environment, and define terminology to be used throughout this book.

This chapter covers the following topics:

- ▶ High availability concepts
- ▶ High availability terminology

1.1 High availability concepts

Any discussion of high availability for IT organizations includes topics such as business requirements, service level agreements, single points of failure, and return on investment. In today's business climate, critical applications must be available 24 hours a day, seven days a week. People expect that retail, banking, government, news, public service, travel, and cloud applications are available all the time, or that they offer high availability; see Figure 1-1.

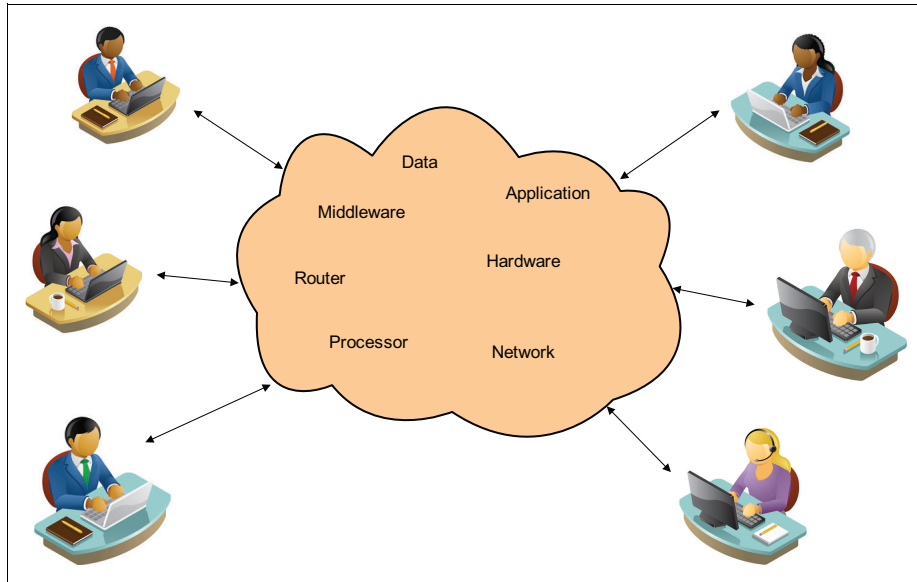


Figure 1-1 Need for high availability

Users of corporate or internet applications view availability in terms of the application being available to them. They do not see, or care about, the hardware and software infrastructure behind the application interface. Each application is made up of many pieces of hardware and software. Data and commands travel from the user's keyboard or touch screen, through communication networks, firewalls, routers, processors, and disk systems.

Applications made up of many hardware and software components are a complex chain with many links. When there is a weak link in the chain, the connection is broken and the application is unavailable. The more links in the chain, the greater the potential for breakage and loss of service to the user.

IT organizations need to consider and implement high availability solutions for critical business applications, but not all applications are critical. When assessing the importance of applications, you must think about the impact on your business when an application has an outage. How much does it cost the business for each minute an application is down, both in terms of today's revenue and for future business growth?

After you identify critical applications that require high availability, you need to look at each hardware and software component required for those applications. What are the weak links in the application chain? Historical outage reports should show any hardware or software components that caused problems in the past. When a weak link is identified, the next step is to see if it can be eliminated by redundant hardware, network, processors, or software upgrades.

Vulnerabilities in the application components cause outages that might adversely affect business; see Figure 1-2 for some of the effects of application outages.



Figure 1-2 Effects of application outage

Many applications and systems do not require continuous availability. But, employees and customers are each ultimately affected by system outages. Business needs and service level agreements should dictate when an application needs a highly available infrastructure, or backup system.

1.2 High availability terminology

Information Technology (IT) resilience is the ability to rapidly adapt and respond to any internal or external disruption, demand, or threat, and continue business operations without significant impact.

IT resilience is related to, but broader in scope, than disaster recovery. Disaster recovery concentrates solely on recovering from an unplanned event. When investigating IT resilience options, two objectives must be at the forefront of your thinking, recovery time and data recovery point.

This section defines some of the technical words and phrases used in this publication to describe the processes, architecture and concepts of high availability.

1.2.1 Availability processes and targets

High availability planning generally revolves around describing the goals for important business processes and applications. The terms defined in this section deal with that part of a high availability discussion.

Availability

There are several definitions of availability but, for the purpose of this book, availability is the degree in which a service or application is ready for use or available (uptime).

High Availability	High availability is when service or application uptime approaches 100%. The terms stated in SLAs determine the target for a system's high availability. A system that is designed to be highly available withstands failures that are caused by planned or unplanned outages. It provides service during defined periods, at acceptable or agreed upon levels, and masks unplanned outages from end-users. It employs Fault Tolerance; Automated Failure Detection, Recovery, Bypass Reconfiguration, Testing, Problem and Change Management.
Continuous availability	Continuous availability is a continuous, nondisruptive, level of service that is provided to users. It provides the highest level of availability that can possibly be achieved. Planned or unplanned outages of hardware or software cannot exist in environments that are designed to provide continuous availability.
Continuous operations	Continuous operations is a continuous, nondisruptive, level of operation where changes to hardware and software are transparent to users. Planned outages typically occur on environments that are designed to provide continuous operation. These types of environments are designed to avoid unplanned outages.
Service level agreement	Service level agreements (SLAs) determine the degree of responsibility to maintain services that are available to users, costs, resources, and the complexity of the services. For example, a banking application that handles stock trading must maintain the highest degree of availability during stock trading hours. If the application goes down, users are directly affected and, as a result, the business suffers. The degree of responsibility varies depending on the needs of the user.
Five 9's or 99.999 availability	High availability service level agreements frequently have a target of 99.999% availability. Mathematically the target of 99.999 means that 0.001% of the time the system or application is unavailable to users. That's about 26 seconds of unavailable time per month, or 5 1/2 minutes per year; see Table 1-1. That is great for many applications, such as background work or report generators, but could be unacceptable for retail websites, banking or credit applications.

Table 1-1 Five nines availability

Availability target	Outage time limit per month
99%	7 hours 12 minutes
99.9%	43 minutes
99.99%	4 minutes 20 seconds
99.999	26 seconds

1.2.2 Outage and recovery terminology

The language of IT outages and recovery processes includes terminology used when describing the situations when the harshness of reality conflicts with IT availability plans. This section includes those types of phrases and keywords used in this publication.

Recovery Time Objective (RTO)

This term refers to how long your business can afford to wait for IT services to be resumed following a disaster. If this number is not clearly stated now, think back to the last time you had a significant service outage. How long was that outage, and how much pain did your company suffer as a result? This will help you get a sense of whether to measure your RTO in days (see Figure 1-3), hours, or minutes.



Figure 1-3 Recovery time delays

Recovery Point Objective (RPO)

This term refers to how much data your company is willing to have to recreate following a disaster. In other words, what is the acceptable time difference between the data in your production system and the data at the recovery site? As an example, if your disaster recovery solution depends on once-daily full volume tape dumps, your RPO is 24 to 48 hours depending on when the tapes are taken offsite. If your business requires an RPO of less than 24 hours, you will almost certainly be forced to perform some form of offsite real time data replication instead of relying on these tapes alone.

The terms RTO and RPO are core to the solution that you choose to meet your business IT resilience needs.

Outage

For the purpose of this book, outage is the loss of services or applications for a specific period of time. An outage can be planned or unplanned.

Planned outage: Occurs when services or applications are stopped because of scheduled maintenance or changes, which are expected to be restored at a specific time.

Unplanned outage: Occurs when services or applications are stopped because of events that are not in our control such as natural disasters. Also, human errors and hardware or software failures can cause unplanned outages.

Uptime

Uptime is the length of time when services or applications are available.

Downtime

Downtime is the length of time when services or applications are not available to users. It is usually measured from the time that the outage takes place to the time when the services or applications are available.

Disaster recovery

Protection against down time caused by unplanned outages such as natural disasters through reliable, predictable recovery procedures.

Single point of failure

A single point of failure (SPOF) exists when a hardware or software component of a system can potentially bring down the entire system without any means of quick recovery. Highly available systems tend to avoid a single point of failure by using redundancy in every operation.

Failover

Failover is the process in which one or more server resources are transferred to another server or servers in the same cluster because of failure or maintenance.

Failback

Failback is the process in which one or more resources of a failed server are returned to its original owner when it becomes available.

1.2.3 HA architecture terminology

In this section, we outline some of the most common terms that are used to describe high availability.

Cluster

A cluster is a group of servers and resources that act as one entity to enable high availability or load balancing capabilities. Figure 1-4 shows an example of a cluster.

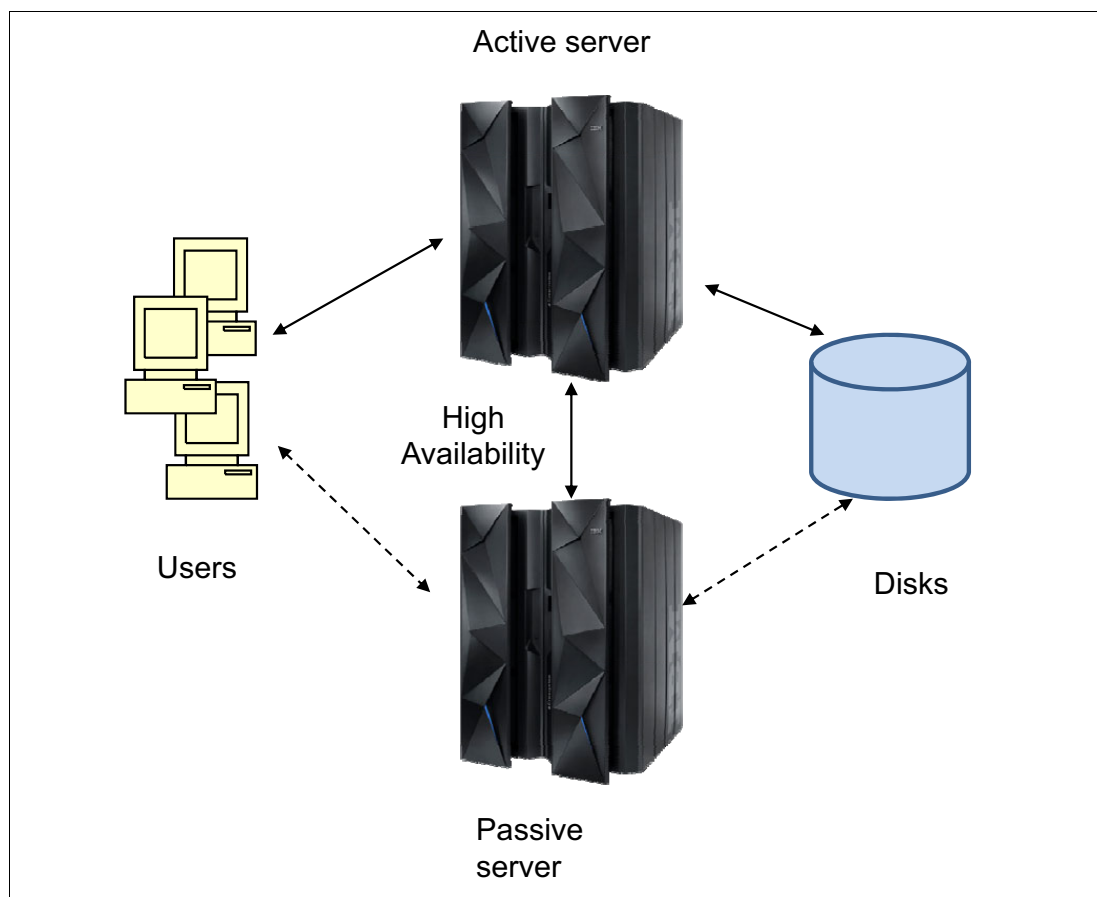


Figure 1-4 An example cluster

Standby (secondary, passive, or failover) server	A standby server, also known as a passive or failover server, is a member of a cluster that is capable of accessing resources and running processes. However, it is in a state of hold until the primary server is compromised or has to be stopped. At that point, all resources fail over the standby server, which becomes the active server.
Split-brain scenario	In a split-brain scenario, more than one server or application that belongs to the same cluster can access the same resources, which in turn can potentially cause harm to these resources. This scenario tends to happen when each server in the cluster believes that the other servers are down and start taking over resources.
Fencing	Fencing is a mechanism used in high availability solutions to block an unstable cluster member from accessing shared resources and communicating with other members or systems. When fencing is applied, the unstable server cannot run any processes until its communication to the other servers in the cluster is resumed. Shoot The Other Node In The Head (STONITH) is one technique that is used to implement fencing (see Figure 1-5).

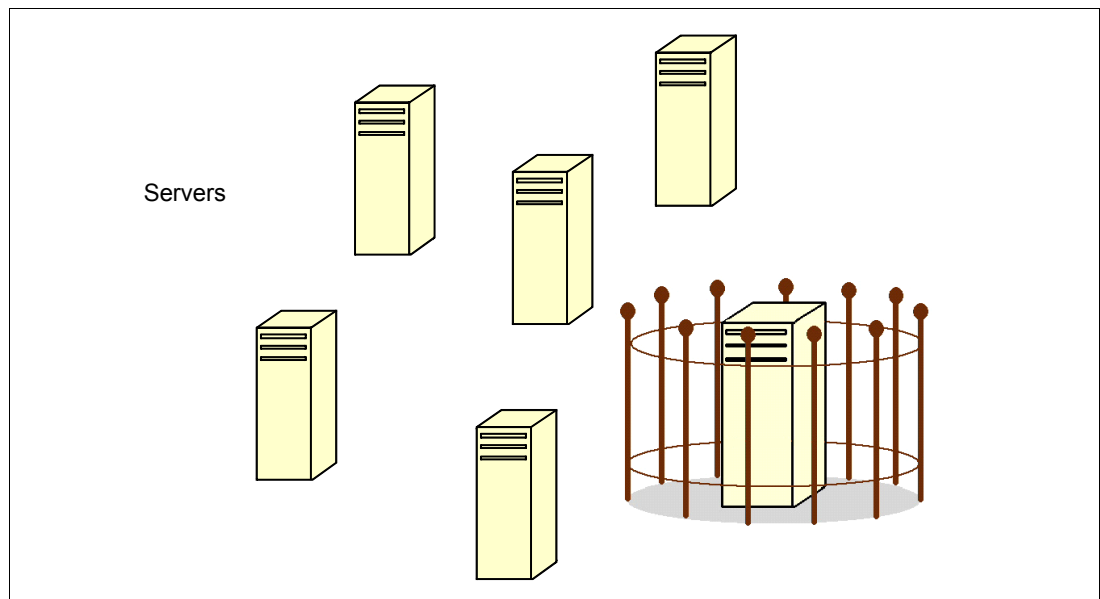


Figure 1-5 Fencing

Quorum	Quorum is a mechanism that is used to avoid split-brain situations by selecting a subset of the cluster to represent the whole cluster when is forced to split into multiple sub-clusters due to communication issues. The selected cluster subset can run services that make the cluster available.
---------------	--

IBM Geographically Dispersed Parallel Sysplex™ (IBM GDPS)

GDPS is a multi-site (as shown in Figure 1-6) or single-site end-to-end application availability solution that provides the capability to manage remote copy configuration and storage subsystems, to automate Parallel Sysplex operation tasks and perform failure recovery from a single point of control.

GDPS is actually a collection of several IBM offerings, each addressing a different set of IT resiliency goals, that can be tailored to meet the RPO and RTO for your business. Each offering leverages a combination of server and storage hardware or software-based replication and automation and clustering software technologies.

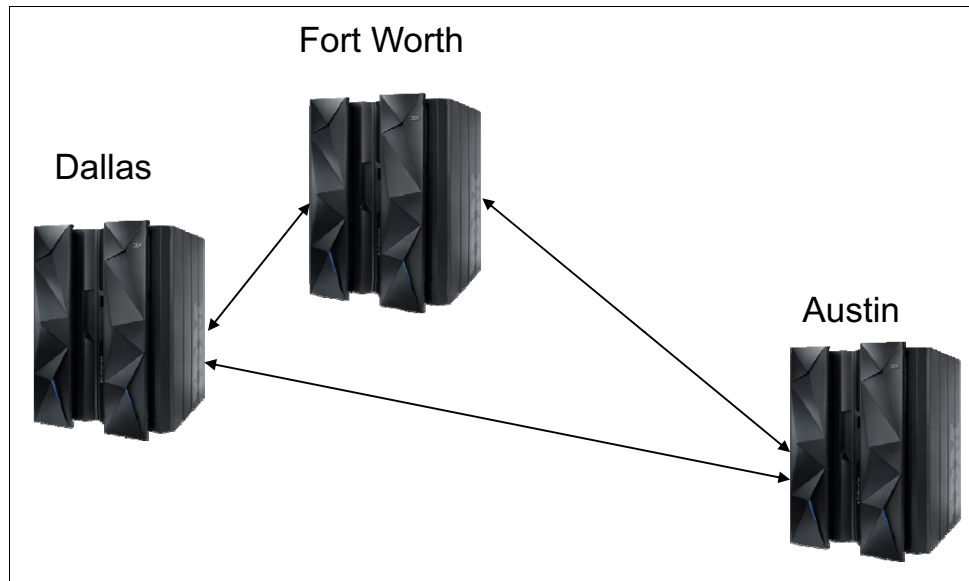


Figure 1-6 Geographically dispersed systems

Active / Passive

An active/passive configuration, has a main system and backup system or site. Resources are configured to run on the main system. When the application is started, only the active site serves the resources or users. The passive systems are running but do not have any resources in production. Upon failure of the active system, HA software transitions the resources to run on one of the passive nodes based on the resource's constraint settings.

Active / Active

With an active/active configuration, the application environment is made up of two or more active systems running resources at the same time. The resources are configured to run on all the sites in the cluster. With active/active configurations you might have a load balancer in front of the cluster to load balance the incoming requests among the active nodes in the cluster. When a system fails, its service workload is migrated to an active node. When one active member fails, the resources are still running on the other active members, and the new incoming service is uninterrupted. Systems must have sufficient hardware resources to handle extra work in case of an outage of one system; or work must be prioritized and service restricted in case of a system failure.

The end-to-end application represented by Figure 1-7 has a pathway from internet users to applications and data on IBM z/OS. This type of application architecture is common in today's business world. The figure illustrates the number of hardware, software and network components that can be part of the pathway. Each piece of the path is a potential failure point.

Critical business applications will need to have redundancy as depicted here to offer high availability. Including multiple paths from the users through communication servers, application servers and database systems improves availability by reducing the likelihood of encountering a single point of failure.

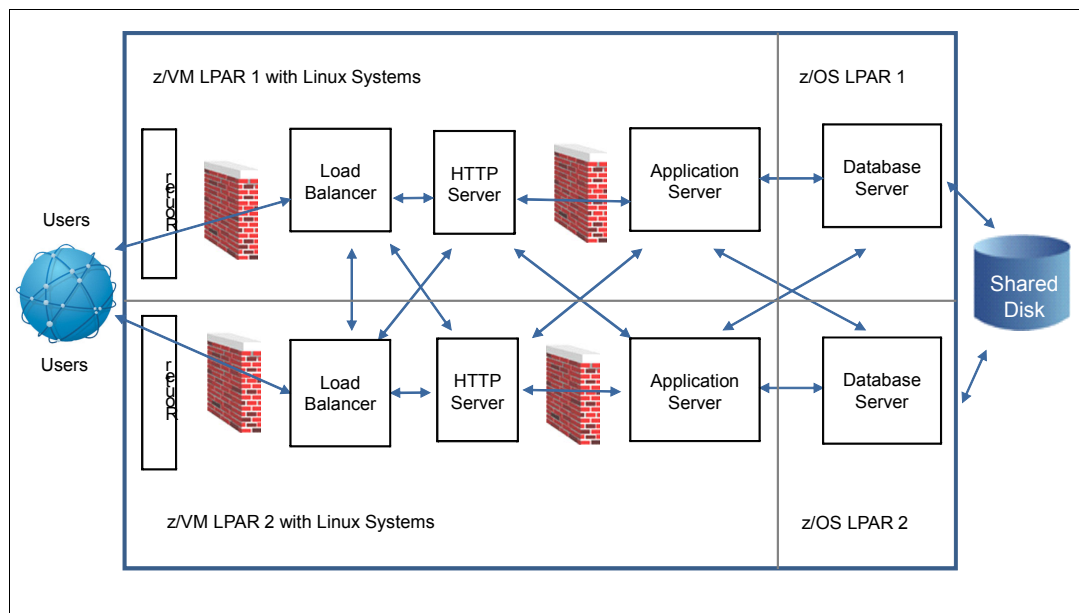


Figure 1-7 Active / Active systems



Business and IT availability

This chapter discusses the topic of business IT resilience and availability from an IT perspective. We also include a general discussion of high availability concepts and terminology, and introduce the terms that will be used throughout the book.

This chapter covers the following topics:

- ▶ Business and IT resilience
- ▶ High availability inhibitors in IT practices
- ▶ Business impact of outages
- ▶ Determining single points of failure
- ▶ High availability options
- ▶ Deciding on a high availability solution

2.1 Business and IT resilience

Businesses today are looking for ways to improve IT service to their customers, simplify operations, and reduce IT costs. Improving service includes increasing availability of applications and their infrastructure or improving IT resilience.

IT resilience can be defined as the ability to rapidly adapt and respond to any internal or external disruption, demand, or threat, and continue business operations without significant impact. IT resilience is the goal when creating a high availability environment. It is broader in scope than disaster recovery. Disaster recovery concentrates on recovering from unplanned events, especially major disasters (as shown in Figure 2-1).



Figure 2-1 Is IT ready for a disaster?

When investigating IT resilience options, two things must be at the forefront of your thinking:

- ▶ Recovery Time Objective (RTO): How long a business can afford to wait for IT services to be resumed following a disaster.
- ▶ Recovery Point Objective (RPO): How much data a company is willing to have to recreate following a disaster. In other words, what is the acceptable amount of time to get critical data to the recovery site?

High availability and disaster recovery solutions both must evaluate RTO and RPO. Disaster recovery (DR) processes are a subset of a company's high availability processes. However, DR is not the focus of this publication. It is important to mention it here as businesses need to consider plans for disaster recovery along with creating a highly available IT environment.

The practice of preparing for DR is something that has been a focus of IT planning for many years. In turn, there is a wide range of offerings and approaches available to accomplish DR. Several options rely on offsite or even outsourced locations that are contracted to provide data protection or even servers in the event of a true IT disaster. Other options rely on in-house IT infrastructures and technologies that can be managed by your own teams.

There is no one correct answer for which approach is better for every business, but the first step in deciding what makes the most sense for you is to have a good view of your IT vulnerabilities. What are the most likely causes of a disaster or outage of your applications. You will also need to assess your resiliency objectives for critical applications; specifically, your RPO and RTO.

In addition to the ability to recover from a disaster, many businesses now look for a greater level of availability covering a wider range of events and scenarios. This larger requirement is called IT resilience. In this IBM Redbooks publication, we concentrate on *high availability* (HA), which encompasses not only recovering from disasters, but keeping your applications up and running throughout the far more common planned and unplanned outages that do not constitute an actual disaster. Some organizations might need to go a step further for business critical applications and provide near-continuous availability.

For a business today, few events impact a company like having an IT outage, even for a matter of minutes (see Figure 2-2), and then finding the incident splashed across the newspapers and the evening news. Today, your clients, employees, and suppliers expect to be able to do business with you around the clock, and from all corners of the globe.



Figure 2-2 Effect of an outage

To help keep business operations running 24x365, you need a comprehensive business continuity plan that goes beyond disaster recovery. Maintaining high availability and continuous operations in normal day-to-day operations are also fundamental for success. Businesses need resiliency to help ensure that these requirements are met:

- ▶ Key business applications and data are protected and available.
- ▶ If a disaster occurs, business operations can continue with a minimal impact.

In this section, we discuss some of the business drivers behind the need for IT resilience, such as these:

- ▶ Increasing regulatory requirements.
- ▶ High and constantly increasing client and market requirements for continuous availability of IT processes

There are several other business drivers behind the need for IT resilience:

- ▶ Financial loss due to lost revenue, punitive penalties or fines, or legal actions that are a direct result of disruption to critical business services and functions
- ▶ An increasing number of security-related incidents, causing severe business impact
- ▶ Major potential business impact in areas such as market reputation and brand image from security or outage incidents.

2.1.1 Regulations

In some countries, government regulations specify how organizations must handle data and business processes. An example is the Health Insurance Portability and Accountability Act (HIPAA) in the United States. This law defines how an entire industry, the US health care industry, must handle and account for patient-related data.

This is also an area that accelerates as financial systems around the world become more interconnected. Although a set of recommendations published in Singapore (such as the SS 540-2008 Standard on Business Continuity Management) might only be directly addressing businesses in a relatively small area, it is common for companies to do business in many countries around the world, where these might be requirements for ongoing business operations of any kind.

2.1.2 Business requirements

It is important to understand that the cost and complexity of a solution can increase as you get closer to true continuous availability, and that the value of a potential loss must be borne in mind when deciding which solution you need, and which one you can afford. You do not want to spend more money on a continuous availability solution than the financial loss you can incur as a result of a outage.

A solution must be identified that balances the costs of the solution with the financial impact of an outage. A number of studies have been done to identify the cost of an outage; however, most of them are several years old and do not accurately reflect the degree of dependence most modern businesses have on their IT systems.

Therefore, your company needs to calculate the impact in your specific case. If you have not already conducted such an exercise, you might be surprised at how difficult it is to arrive at an accurate number. For example, if you are a retailer and you suffer an outage in the middle of the night after all the batch work has completed, the financial impact is far less than if you had an outage of equal duration in the middle of your busiest shopping day. Nevertheless, to understand the value of the solution, you must go through this exercise, using assumptions that are fair and reasonable.

2.1.3 Other considerations

In addition to the increasingly stringent availability requirements for traditional mainframe applications, there are other considerations, including those described in this section. Key areas to consider have changed over the years and now include increasing complexity of applications and data, high client expectations, and global presence required for businesses.

Increasing application complexity

The mixture of disparate platforms, operating systems, and communication protocols found within most organizations intensifies the already complex task of preserving and recovering business operations. Reliable processes are required for recovering not only the mainframe data, but also perhaps data accessed by multiple flavors of Linux, UNIX, Microsoft Windows, or even a proliferation of virtualized distributed servers.

It is becoming increasingly common to have business transactions that span, and update data on, multiple platforms and operating systems. If a disaster occurs, your processes must be designed to recover this data in a consistent manner. Just as you would not consider recovering half an application's data to 8:00 a.m. and the other half to 5:00 p.m., the data touched by these end-to-end applications must be managed to ensure that *all* this data is recovered with consistency to a single point in time. The exponential growth in the amount of data generated by today's business processes and IT servers compounds this challenge.

Increasing infrastructure complexity

A large business today will have multiple data centers. Each data center has a variety of platforms, hardware types, operating systems, and networks. Mainframe systems comprise only a small part of the equipment in a data center, but often manage the most vital data. When an outage or disaster occurs, how confident are you that all those platforms can be recovered? And if they can be recovered, will the data from each system be in synch and reset to the same point in time as your mainframe systems? Another consideration is the recovery time for restoring the data that is important for reestablishing business processes?

Figure 2-3 shows a typical IT infrastructure. If you have a disaster and recover the mainframe systems, will you be able to recover your service without all the other components applications to be available, so that users can access them? Therefore, part of your IT resilience solution must include not only addressing the non-mainframe parts of your infrastructure, but also ensuring that recovery is integrated with the mainframe plan.

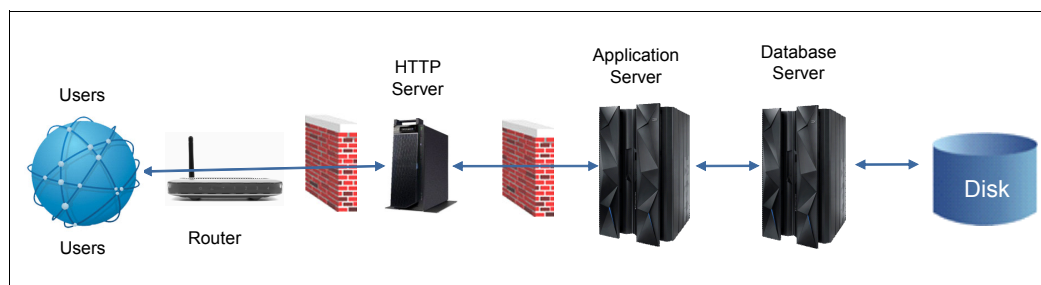


Figure 2-3 End-to-end architecture

IT resilience encompasses much more than the ability to get your applications up and running after a disaster with “some” amount of data loss, and after “some” amount of time. When investigating an IT resilience solution, keep the following points in mind:

- ▶ **Planned server and site outages**

Does the proposed solution provide the ability to stop a system in an orderly manner? Does it provide the ability to move a system from the production site to the backup site in a planned manner? Does it support server clustering, data sharing, and workload balancing, so the planned outage can be masked from users?

Does the proposed solution provide the ability to move the entire production environment (systems, software subsystems, applications, and data) from the production site to the recovery site? Does it provide the ability to move production systems back and forth between production and recovery sites with minimal or no manual intervention?

► Managing end-to-end data replication

Does the HA solution support data from more systems than just z/OS? Does it provide data consistency across all supported platforms, or only within the data from each platform?

Does the solution provide an easy-to-use interface for monitoring and managing the data replication environment? Will it automatically react to connectivity or other failures in the overall configuration?

Does the solution provide consistency across all replicated data? Does it provide support for protecting the consistency of the second copy if it is necessary to resynchronize the primary and secondary copy?

► Support for continuous application availability

Does the solution support continuous application availability? From the failure of any component? From the failure of a complete site?

► Support for processor failures

Does the solution support recovery from a processor failure? Is the recovery disruptive (reboot / re-IPL) or transparent (IBM HyperSwap®, for example)?

► Automated recover and dynamic provisioning of resources

Does the solution have the ability to dynamically allocate resources and manage workloads? Will critical workloads continue to meet their service objectives, based on business priorities, in the event of a failure? Can recovery be automated?

► Support for recovery from regional disasters

Does your business currently have, or plan to create regional data centers (Figure 2-4)? What distances are supported by the HA solution? What is the impact on response times? Does the distance required for protection from regional disasters permit a HA application capability?

Evaluation of availability targets must start with the applications that are needed to keep your business running every minute of every day. The next category of business applications in order of importance has less urgency for a HA solution, and some systems do not have to be available each day. You then need to compare your company's requirements in each of these categories against your existing or proposed solution for providing IT resilience.

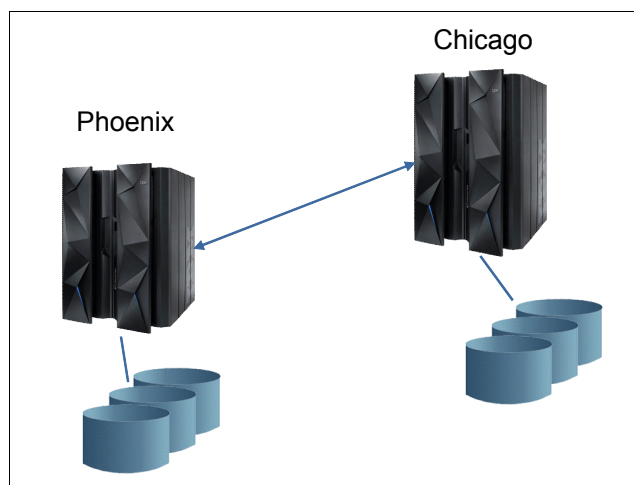


Figure 2-4 Regional data centers

2.2 High availability inhibitors in IT practices

As noted earlier in this chapter, companies increasingly depend upon websites as part of their standard business operations. Thus, when a website goes down, it can cost the business money, and getting a website back up quickly is essential. However, practices that a company's IT organization perform each day can sometimes unintentionally keep the company from meeting this important goal.

Thorough planning, quality control, and change management are essential to improving application availability. Conversely, the lack of process management in those areas can sometimes prove problematic for the IT infrastructure and applications. Creating good availability processes, such as creating good applications, requires management through four phases:

1. Planning
2. Development
3. Validation
4. Implementation and updating

In addition to the mistakes listed in Table 2-1, common to all phases is the possibility of a communication breakdown. Communication is critical in today's business environment where changes occur rapidly. Key to success is clear communication between people and across organizations. Miscommunication can be frustrating and can lead to outages or long recovery times.

Table 2-1 HA process creation mistakes

HA Process phase	Common inhibitors to success
Planning	<ul style="list-style-type: none">▶ Outdated architecture diagram▶ No plan to education staff▶ No capacity or scalability plan
Development	<ul style="list-style-type: none">▶ Lack of documentation▶ Poor management of the end-to-end process design
Validation	<ul style="list-style-type: none">▶ Test environment not equal to production▶ No load or stress testing▶ Changes put directly into production
Implementation and updating	<ul style="list-style-type: none">▶ No periodic testing▶ Processes stagnate after initial implementation

In this section, we discuss these four phases.

2.2.1 Phase 1: Planning for high availability

When the IT department starts the process to improve availability and move their critical business systems toward high availability, the first stage is planning. Good planning lays the foundation for achieving the goal of highly available systems.

Having an architecture plan that is current can help you to understand how a change in one component affects the other components. This plan allows your organization to upgrade applications and diagnose issues more easily. When no architecture plan is in place, an HA solution might be based off of an old architectural diagram and hardware component selections may not meet requirements.

Part of the planning stage determines the amount of data that will flow into, through, and out of the application. The type of data and the volume of that data will grow over time. If a capacity or scalability plan is not provided for, organizations often encounter a lack of needed information:

- ▶ No prediction for the needed production capacity or response time
- ▶ No list of which hardware components are used, and how often
- ▶ No prediction of how many concurrent users will make up the production workload
- ▶ No periodic update to the plan for seasonal changes and business growth

The planning stage also includes allowing time for the organization to learn the requirements of and how to effectively use and tune any applications. Time and resource constraints can prevent formal class attendance, or organizations might not allocate time to study educational materials, both of which results in limited knowledge on new high availability processes.

2.2.2 Phase 2: Development

The development phase is the creation stage, which builds a foundation for the production environment. This stage is more than just the original development of the high availability infrastructure. It also includes installation of all of the modifications to the current environment and perhaps building new data center space.

During the development phase, as well as the others, it is important to have good project design, planning and management. Communication throughout the stages will help ensure a quality solution and perhaps identify the need for solution design changes as the business evolves.

2.2.3 Phase 3: Validation

The validation phase is a testing phase that verifies that the high availability solution works as desired in a complex environment. The environment is similar to your production environment in terms of the network, hardware configuration, back-end database, and load.

When designing highly available applications on heterogeneous platforms, the validation stage, can be lengthy. The most common mistakes that cause issues in the solution are brought about by lack of project management or inadequate time for testing.

Often, there is not enough budget to create a test environment that is equal to the production environment (Figure 2-5).

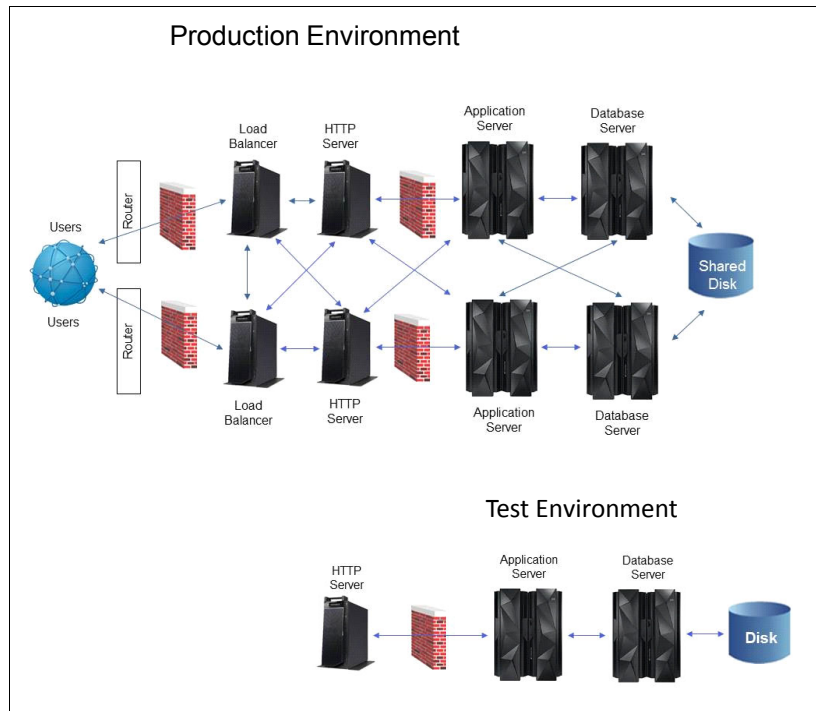


Figure 2-5 Production versus test environment

The test environment may be too small or is overcommitted and not available when it is needed. The test hardware, network, or software levels might differ from the production environment; or the z/OS logical partition (LPAR) or network are not the same as production.

When time allotted for the validation phase is short it is impossible to properly test the high availability solution components. It is important to test a failure for each component of an end-to-end application to be sure that there are no single points of failure, or that the impact of a failure is minimized. Also, a short window for testing can lead to changes put directly into production without proper validation procedures.

2.2.4 Phase 4: Implementation and updating

The implementation and updating phase is the key stage for the lifecycle of the HA environment and is far longer than the original application development stage. However, it is possible to overlook or forget about this phase. Note, however, that at this stage, the HA environment is now in production. Thus, fixing problems is often costly and visible to the public.

The software and hardware platform that applications are built upon evolve over time. Thus, co-requisite components need upgrades and integration testing. Hardware and operating systems might be upgraded, and then the application might expand or interface with additional applications or databases. All these changes require adequate planning, testing, documentation updates, and staff training. Planning for high availability, like most IT systems, will change and must be updated as shown in Figure 2-6.

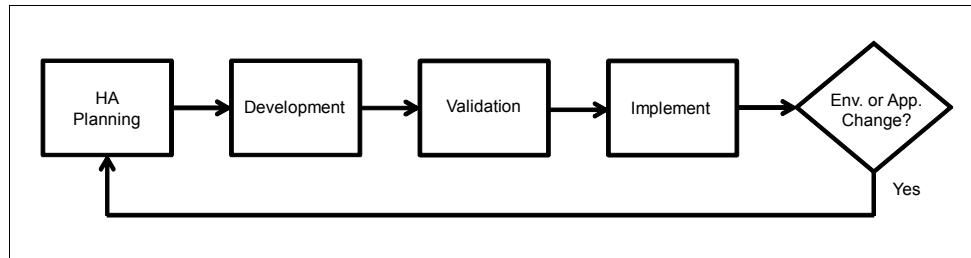


Figure 2-6 HA implementation and update cycle

When a concurrent record of changes is not made, documentation becomes worthless. This can happen when changes are made under stress, or when different teams make changes to the environment without good change management processes and without communicating well with each other.

2.3 Business impact of outages

Even as the demand for continuous availability increases, it is still difficult for our clients to justify the expense of implementing continuously available systems and procedures. General figures can be quoted, but applying this to individual business applications is difficult. Sources of outage costs include loss of revenue or sales through e-commerce, regulatory fines for downtime or deadlines missed, customer compensation, reputation damage that leads to lost sales, lost productivity for employees who need to use systems, or lost productivity for employees who manage the data center.

The cost of an unplanned and even planned outage is often cited only during an unplanned outage or after a recent outage. Even if the cost per minute (or other metric) of an outage can be calculated accurately, it is difficult to predict how many outages will occur in a year or how long each might last.

In this section, we discuss how to quantify the cost of making services continuously available, the differences between continuous availability, high availability, and disaster recovery, a heterogeneous end-to-end view, and the importance of communication between the business and IT.

2.3.1 Quantifying the cost of making services continuously available

As it is typically approached by organizations, disaster recovery (DR) is a regulatory or other requirement (even though not always implemented correctly). The accepted cost of DR can be compared to the cost of high availability, where the hardware and software architecture is duplicated in a distant data center and DR procedure exercises are conducted occasionally.

Figure 2-7 demonstrates the decisions involved in planning for disaster recovery; with each decision comes a set of costs associated with that decision.

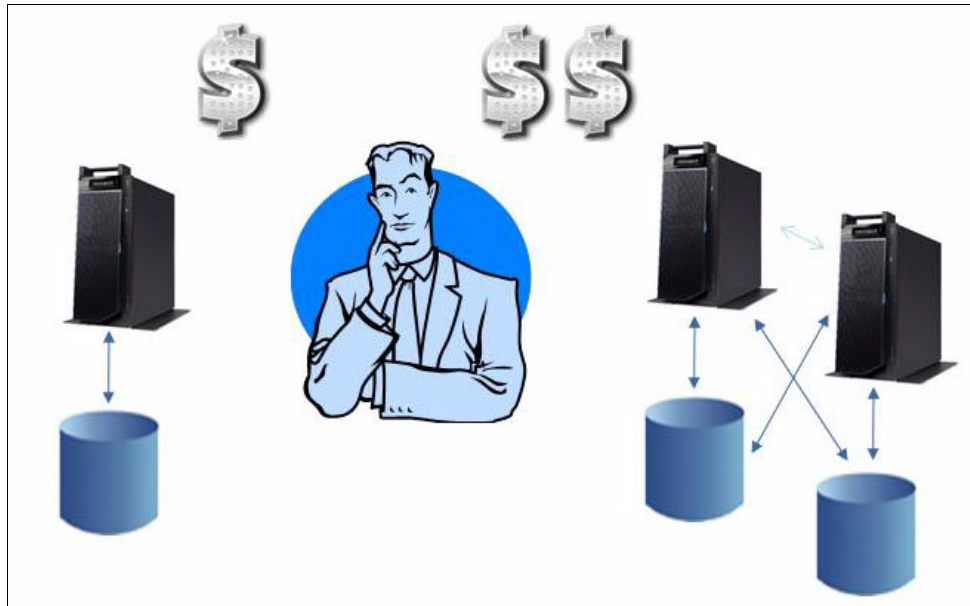


Figure 2-7 Cost of high availability

The main disadvantage of DR solutions is that cost is allocated to idle solutions. Continuous availability is not a regulatory requirement, currently. However, we can envision a differently stated set of regulatory requirements that would require continuous availability. In a continuously available environment, cost is allocated to an active solution. Regardless, decision-makers in organizations typically think that continuous availability is cost-prohibitive.

2.3.2 Continuous availability versus high availability versus disaster recovery

To justify the cost of continuous availability, a measurable impact on the business must be calculated. But often, as soon as the outage is past or if a data center has been fairly stable, the business side forgets the critical need for continuous availability until the next outage occurs. Many businesses seem to understand the impact of a disaster and have disaster recovery plans in place, but they cannot justify continuous availability despite the fact that an unplanned outage is much more likely to occur than an actual disaster.

An organization sometimes even considers the disaster recovery solution to be the solution to a smaller-scale outage. This is problematic because *disaster* has a specific meaning in this context, such as a natural disaster or civil disruption. When an outage occurs on one piece of equipment or an application, affecting one line of business, organizations are reluctant to “call a disaster” (because it is not a disaster) and do the all-or-nothing failover, instead. Understandably, this incurs too much risk for the business.

2.3.3 Heterogeneous end-to-end view

Outages are not caused only by technology (failed hardware, software bugs, and so on). End-to-end continuous availability also includes business architecture, operations, process, and support organizations. Taking an end-to-end view of availability is challenging to organizations that operate with “silos,” or segmented rather than interconnected workgroups, and possibly have conflicting goals. In addition, some clients have multiple strategic outsourcing vendors (based on data center locations, mainframe or distributed systems, for example), and they have deployed software and hardware products from several vendors across the stack: storage, network, database, middleware, and so on.

These heterogeneous environments result in multiple integration points and complex interactions. Determining what portions of the environment require continuous availability and coordinating continuous availability across multiple environments is a challenge both technically and organizationally. For our clients, it is not enough that IBM tackles continuous availability. They also need their other vendors to tackle continuous availability, which creates an impetus and need for industry standards.

2.3.4 Poor communication between business and IT

Even when continuous availability for a particular business is a requirement, the recognition of that need is frequently triggered by an unplanned outage. Often, the business managers seem surprised that the outage and resulting loss of service occurred. There can be a disconnect between business and IT, where business promises continuous availability to its clients even without mentioning “continuous availability.” For example, a bank recently launched a marketing campaign that promised mobile banking services anywhere and anytime. Does the marketing department recognize the cost or complexity of keeping this promise? Is the continuous availability requirement communicated to IT and backed up by IT?

There is a need to improve communication between the business organization and the IT organization and to employ terminology and metrics that are meaningful to both and provide a way to communicate continuous availability goals. There is also a need for active and visible leadership from the chief information office.

2.4 Determining single points of failure

To have a high availability configuration in any IT environment, you should avoid any possible single point of failure (SPOF). When applications span platforms in an end-to-end configuration, the number of possible points of failure increases. If high availability is your goal, you should do the planning for eliminating single points of failure early in your project so you can include the requirements for investments in additional hardware and software.

These backup hardware and software components should be tested in a configuration where the primary units are removed to ensure they truly do eliminate the single point of failure in your environment. Testing all possible failure points improves availability and IT resilience.

Consider the end-to-end application shown in Figure 2-8. Looking at each piece of the configuration from the Users to the Mainframe, you can see there are a number of potential failure points. Some pieces have a higher level of availability built into them than others. That might be why they are purchased in the first place. One method of approaching the task of improving availability of an application is to find the weakest links or pieces of the configuration and focusing on each one to improve its availability through duplexing or other means.

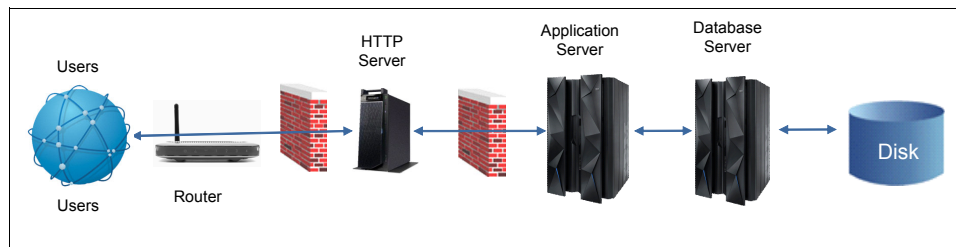


Figure 2-8 End-to-end application

In your own environment, you can look at reports for outages to find those weak links. If, for example, network connectivity is a frequent issue, look at your options for fixing the problem areas that caused outages.

2.5 High availability options

The most common configurations in highly available environments are the active/active configuration and the active/passive configurations. Several factors influence the selection of HA options.

The first factor is the business need or the evaluation of outage tolerance. Is a short outage of a few minutes acceptable for critical applications? Can other applications be unavailable for longer periods of time?

A second factor in choosing high availability options is cost. How much does it cost to create and maintain a highly available solution such as an active / active set of data centers? Businesses always need to evaluate the return on their investment. Yes, highly available solutions are desirable, but can their cost be justified in terms of business need?

2.5.1 Active / active configuration

With an active / active configuration, all servers in the cluster can simultaneously run the same resources. That is, these servers own the same resources and can access them independently of the other servers in the cluster. After a server in the cluster is no longer available, its resources are available on the other servers in the cluster.

An advantage of this configuration is that servers in the cluster are more efficient because they can all work at the same time. However, there is a level of service degradation when one server must run the resources of the server that is no longer in the cluster. In Figure 2-9, both sets of servers have access to the cluster resources and provide services to the users.

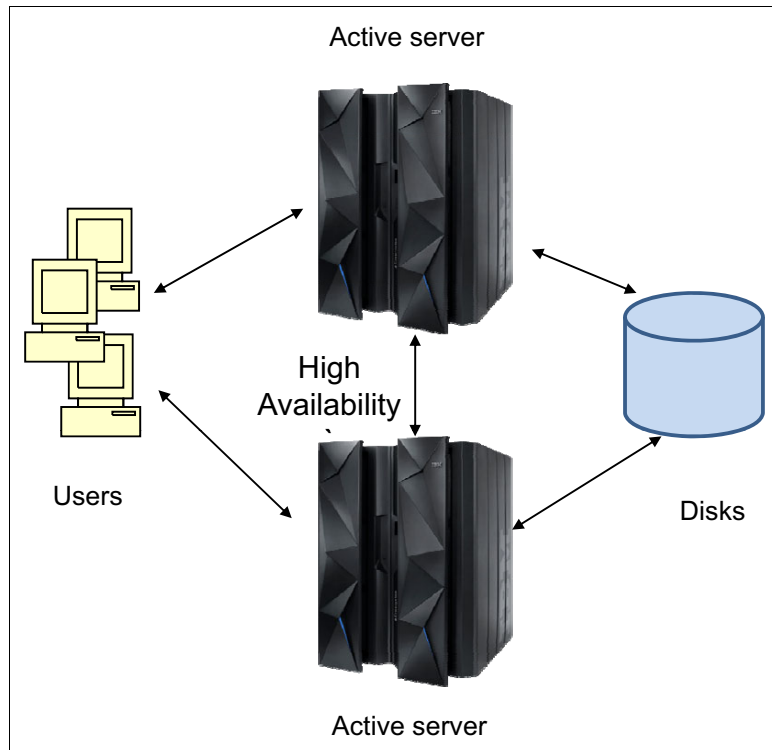


Figure 2-9 Active / active configuration

2.5.2 Active / passive configuration

An active/passive configuration (Figure 2-10) consists of a server that owns the cluster resources and other servers that are capable of accessing the resources that are on standby until the cluster resource owner is no longer available.

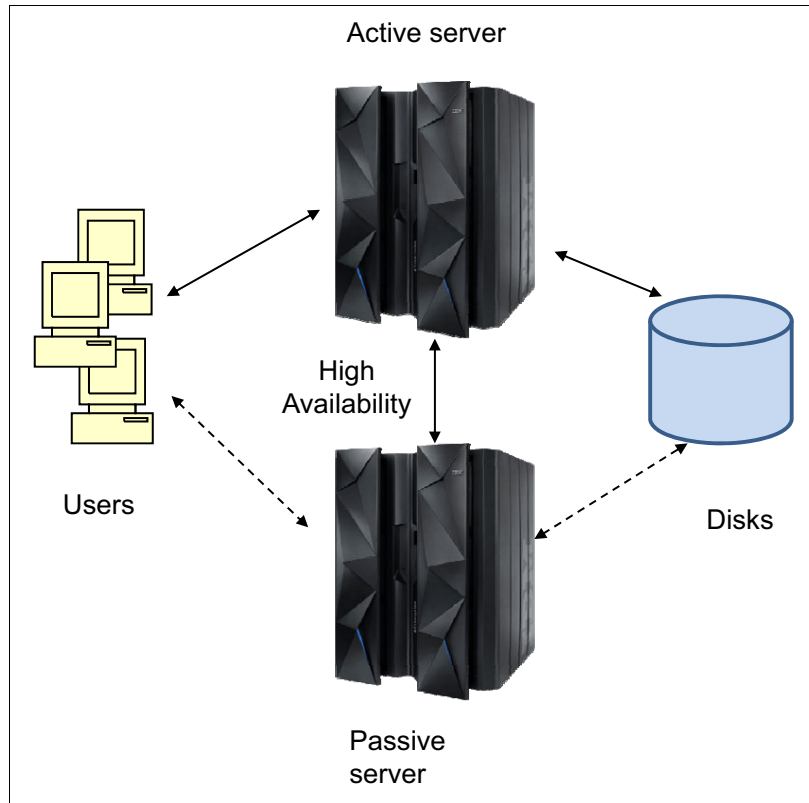


Figure 2-10 Active / passive configuration

The advantages of the active/passive configuration are that there is no service degradation and services are only restarted when the active server no longer responds. However, a disadvantage of this configuration is that the passive server does not provide any type of services while on standby mode, making it less efficient than active/active configurations.

Another disadvantage is that the system takes time to failover the resources to the standby node. In Figure 2-10, the servers shown on the left have access to the cluster resources and provide services to the set of workstations shown at the top of the figure. The servers to the right are on standby and are ready to resume work when indicated. However, they do not provide services to the workstations while the active servers are running.

2.5.3 Being realistic about the cost of continuous availability

The cost of implementing continuous availability initiatives can create a high hurdle (especially the geographically dispersed active / active approach, sometimes called Active-Active-Active or 3-Active). At the same time, we have observed organizations with unrealistic expectations about implementing continuous availability “on the cheap,” given the low price of commodity hardware. Simply throwing hardware at the problem and building a custom solution is a recipe for high maintenance and unattainable SLAs.

The reality is that continuous availability is much more than just hardware and software. The facility costs and resource requirements greatly exceed the initial implementation costs. As a result, continuous availability needs to be viewed in terms of total cost of ownership (TCO), and it must be cost-justified based on the inherent risks of *not* providing continuous availability. Although many believe that cost of the 3-Active approach is exorbitant, that is an invalid analysis for any organization that manages an unused DR site. Many organizations that implement DR without using this capacity can optimize their approach in terms of both cost and resiliency by adopting a 3-Active architecture to optimize continuous availability, with a focus on balancing absolute data consistency requirements with eventual data consistency patterns.

2.6 Deciding on a high availability solution

There are many decisions to make during planning for high availability. The decision tree shown in Figure 2-11 demonstrates the basic process for planning a high availability solution.

Business requirements are the key inputs when considering an HA solution. Review the requirements for high availability with your business department and then decide whether or not to start to plan for HA. For example, you need to determine if there are critical applications identified or if there are business needs for high availability.

There are two high availability indexes, Recovery Time Objective (RTO) and Recovery Point Objective (RPO). All HA solutions can be described using RTO and RPO. However, RPO and RTO are not just simple Yes or No answers to some availability questions. RPO and RTO have to be thoroughly evaluated before deciding on an HA solution. Also keep in mind that the higher the requirements for availability are, the higher the cost.

After a solution is architected, the cost can be evaluated. The solution needs to be reviewed, taking additional aspects into account, such as cost, impact on personnel, and organization. The review may highlight some desired changes to the architected high availability solution.

It is important to keep in mind that there is no single best solution. All methods have their advantages and drawbacks, which need to be considered and a balance found for the most suitable solution, taking into consideration all technical and non-technical aspects.

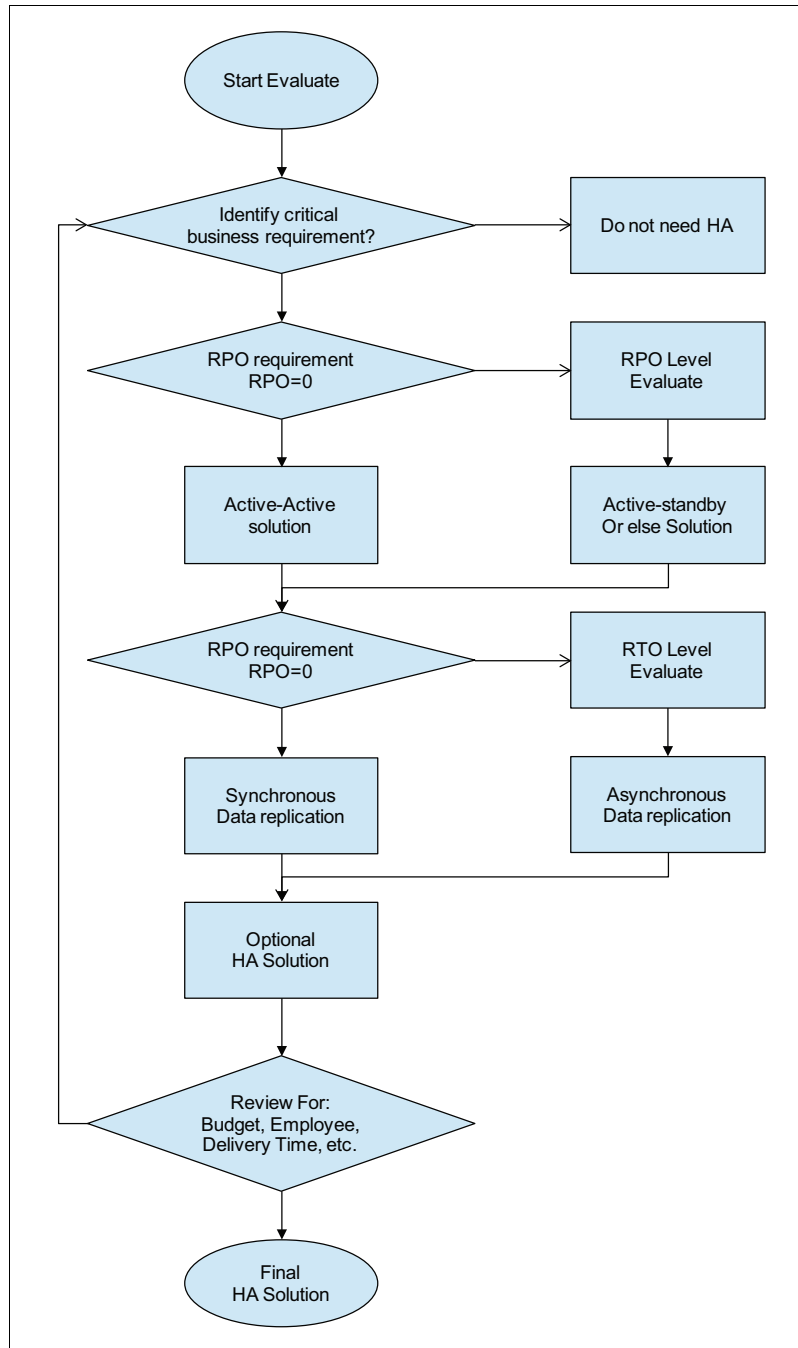


Figure 2-11 How to decide on a high availability solution



Components of high availability

In this chapter, we discuss several different components of a high availability solution and discuss the different technologies that can be used to make the solution highly available. These technologies are divided into the following groups:

- ▶ Processor and storage hardware
- ▶ Storage mirroring and management solutions
- ▶ Operating system HA options
- ▶ Network HA options

This chapter covers the following topics:

- ▶ Components of a high availability solution
- ▶ Hardware high availability
- ▶ Storage high availability
- ▶ Operating system high availability
- ▶ Software solutions for high availability
- ▶ Network high availability

3.1 Components of a high availability solution

When designing a high availability solution, it should be remembered that every aspect of the overall solution can be a potential single point of failure (SPOF). Single points of failure (SPOFs) can be an individual piece of a component such as hardware equipment or software, which will cause system downtime or data loss if it fails.

Major components of a high availability solution are shown in Figure 3-1.

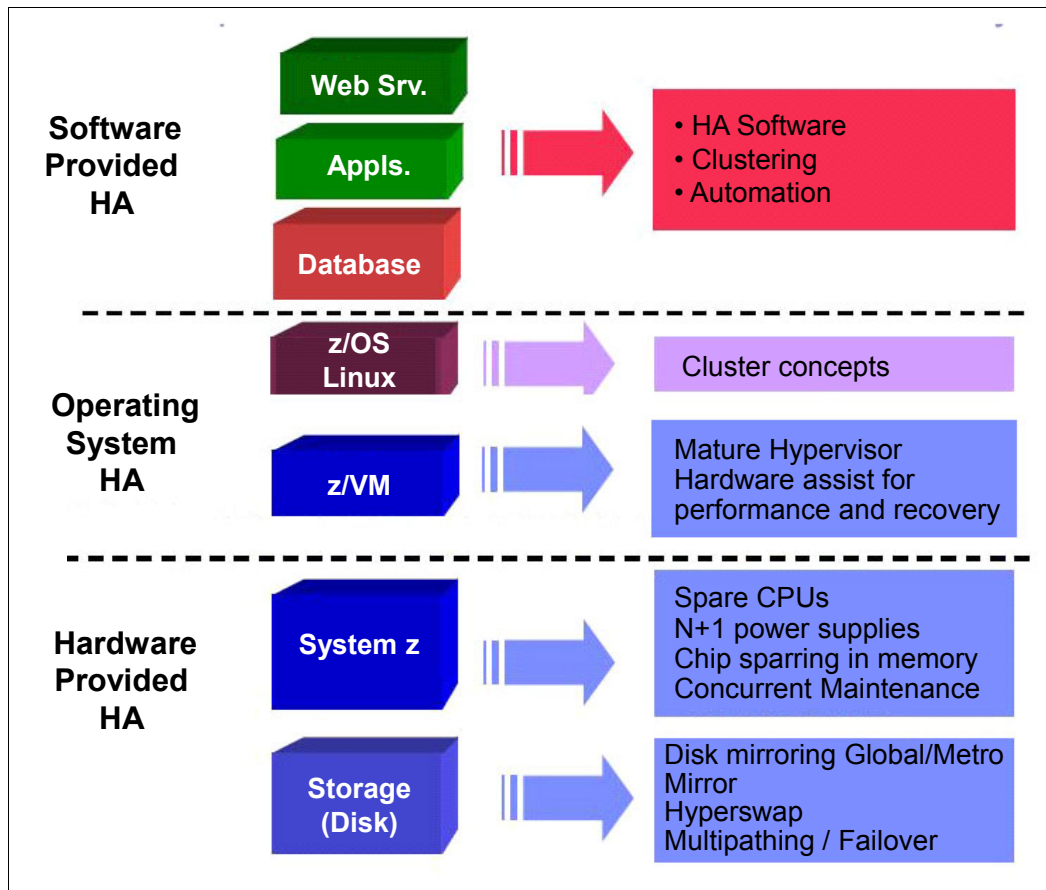


Figure 3-1 Components of High Availability

In the following sections, we discuss each of the components and how various technologies can be used to achieve high availability.

3.2 Hardware high availability

Even the most sophisticated software cannot produce a highly available system without the possibility of failure on a hardware level. This section describes the high availability design of IBM hardware products (System z server and IBM DS8000® Series).

3.2.1 System z hardware

The mainframe (System z server) is one that has been tuned to perfection over the past five decades with built in autonomic capabilities (Self Healing, Self Recovering, Resilient). Hence a single System z machine by itself provides high availability and redundancy, not just by component duplication, but by intelligent sparing. The system comes with error checking and correction components implemented at the hardware level, by way of components and microcode, which gives it the Mean Time Between Failure (MTBF) characteristic of several decades.

The following key hardware components should be considered and laid out with the greatest possible redundancy for the server:

- ▶ Power Supply
- ▶ Processor Sparing
- ▶ Memory

Power

Mainframes operate with two sets of redundant power supplies. Each set of the power supplies has its individual power cords or pair of power cords, depending on the number of Bulk Power Regulator (BPR) pairs installed. Apart from the power supplies redundancy, there is an optional feature to have inbuilt battery support. The Integrated Battery Feature (IBF) enhances the robustness of the power design, increasing power line disturbance immunity. The feature provides battery power to preserve processor data if there is a total loss of power from the utility company. The IBF can hold power briefly during a brownout, or for orderly shutdown in a longer outage.

Processor Sparing

Processor Sparing is an error detection and correction mechanism in the mainframe, which allows error detection during instruction execution and transparent error correction of spare processors have been configured to the system.

In the event of faulty processor, the Service Assist Processor (SAP) activates the spare processor and moves all register and cache contents to the new CPU. In addition the CPU id from the failing CPU is also moved to the spare CPU so that the new CPU looks identical to the failing CPU. The newly configured CPU can now immediately start to process the failing instruction again and can seamlessly and transparently replace the failing CPU.

Memory

In System z, for enhanced availability, memory is implemented as a Redundant Array of Independent Memory (RAIM). The RAIM subsystem is a revolutionary, next-generation memory design that tolerates complete failure of any single DIMM. In the event of failure of single DIMM, the design isolates the faulty DIMM without affecting the system. This design yields the highest level of memory availability of any System z.

The following features were designed to work together to enhance system availability for continuous operations:

Dynamic oscillator switch-over	The zEC12 has two oscillator cards, a primary and a backup. During a primary card failure, the backup card is designed to transparently detect the failure, switch-over, and provide the clock signal to the system.
---------------------------------------	--

Enhanced book availability (EBA)	EBA is a procedure under which a book in a multi-book system can be removed and reinstalled during an upgrade or repair action with no impact on the workload.
---	--

Enhanced driver maintenance (EDM) One of the greatest contributors to downtime during planned outages is LIC driver updates that are performed in support of new features and functions. The zEC12 is designed to support the concurrent activation of a selected new driver level.

Note: Concurrent actions such as hardware upgrades, parts replacement, and driver upgrades, which are available for EC12, only work when properly configured for redundancy. For more information, see the *IBM zEnterprise EC12 Technical Guide*, SG24-8049.

3.2.2 IBM DS8000 series

The IBM DS8000 series is designed to help address the needs of on demand environments that require the highest levels of availability. It is designed to support dynamic system changes, such as online system microcode updates and online hardware upgrades. The DS8000 also features redundant, hot-swappable components to help support continuous operations. All disks are RAID-protected, such that multiple spare disks are configured in a RAID group to allow a failed, RAID-protected disk to be rebuilt quickly and automatically to maintain access to information. In this section, we discuss the power subsystem and disk system of the DS8000 series.

Power system

The power subsystem in DS8870 was redesigned from the previous generation of DS8000 series. It offers higher energy efficiency, lower power loss, and reliability improvements, including redundant design.

The following is a list of components that have a redundant power supply:

- ▶ DC-UPS
- ▶ CEC power supply
- ▶ I/O drawer power supply
- ▶ Disk enclosure power supply
- ▶ Rack Power Control Card
- ▶ Power cord
- ▶ Power Junction Assembly for HMC, Ethernet Switch

The following list describes some newly designed features:

- ▶ DC-UPS has replaced the former primary power supply (PPS) and has a built-in power converter capable of power monitoring and integrated battery functions. It distributes full wave rectified AC (or DC voltage from batteries) to Power Distribution Units (PDU), which then provide that power to all the separate areas of the machine.
- ▶ Power Junction Assembly (PJA) is a new element that is introduced in DS8870 power subsystem. Dual PJAs provide redundant power to HMC, Ethernet switches, and HMC tray fans.
- ▶ The BSM set provides backup power to the system when the input AC power is lost. Each DC-UPS supports one or two BSM sets. As standard, there is one BSM in each DC-UPS. If the ePLD feature is ordered, then one BSM set is added. As a result, each DC-UPS has two BSM sets. All racks in the system must have the same number of BSM sets, including expansion racks without I/O.

Disk system

Redundant Array of Independent Disks (RAID) is an industry-wide implementation of methods to store data on multiple physical disks to enhance the availability of that data. The DS8870 supports RAID 5, RAID 6, and RAID 10.

Redundancy path is designed per disk. Figure 3-2 shows the redundancy features of the DS8870 switched Fibre Channel disk architecture. Each disk has two separate connections to the backplane. This configuration allows the disk to be simultaneously attached to both FC switches. If either disk enclosure controller card is removed from the enclosure, the switch that is included in that card is also removed. However, the FC switch in the remaining controller card retains the ability to communicate with all the disks and both device adapters (DAs) in a pair. Equally, each DA has a path to each switch, so it also can tolerate the loss of a single path. If both paths from one DA fail, it cannot access the switches. However, the partner DA retains connection.

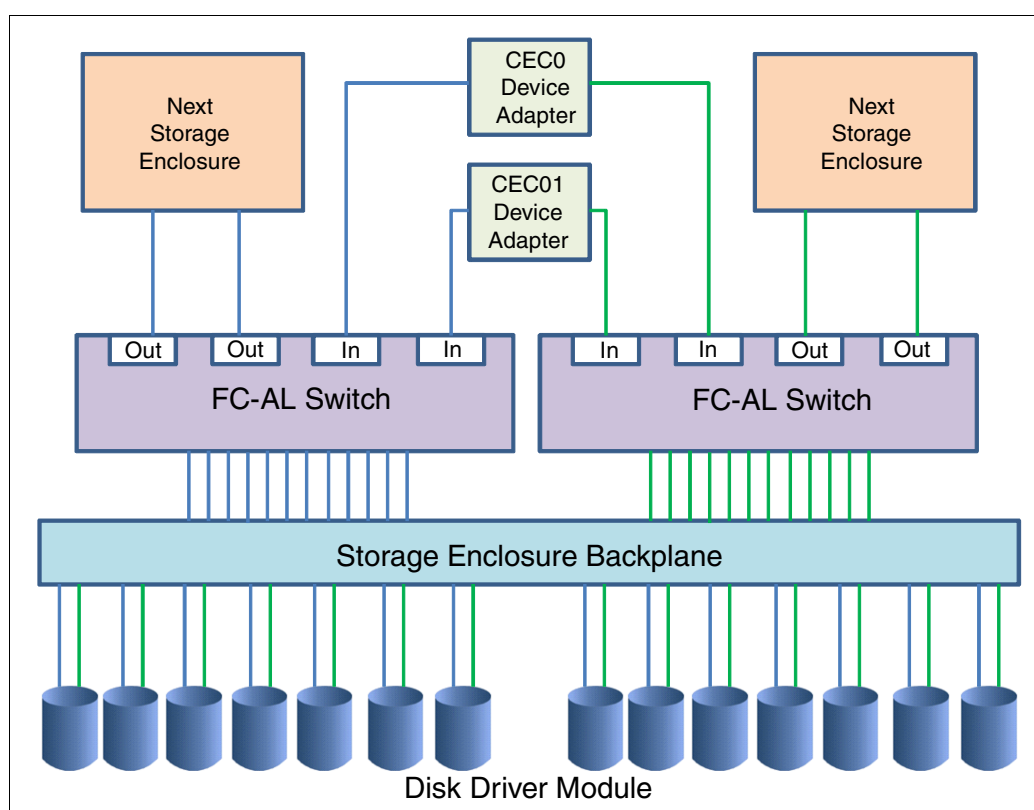


Figure 3-2 Switched Disk Path Connection

Smart Rebuild is a feature that is designed to help reduce the possibility of secondary failures and data loss in RAID arrays. Smart Rebuild disk drive module (DDM) error patterns are now continuously analyzed in real time as part of one of the normal tasks that are driven by DS8870 microcode.

At any time, when certain disk errors (following a specific criteria) reach a determined threshold, the Device Adapter (DA) microcode component starts Smart Rebuild immediately. A spare disk is brought into the array. The suspect disk drive and the new spare are set up in a temporary RAID 1, allowing the troubled drive to be duplicated onto the spare rather than performing a full RAID reconstruction from data and parity. After duplicated, the spare disk will be made a regular member of the array and the suspect disk can be removed from the RAID array. The array never goes through an n-1 stage in which it would be exposed to complete failure if another drive in this array encounters errors.

3.3 Storage high availability

Data is the key asset to many businesses, and supporting high data availability is essential. In this section, we describe the storage copy services available for disaster recovery, data migration, and data duplication to meet high availability needs. We also describe some replication options.

3.3.1 Storage copy services

Copy Services are a collection of functions that provide functionality for disaster recovery, data migration, and data duplication solutions. There are two primary types of Copy Services functions, Point-in-Time Copy and Remote Mirror and Copy. Generally, the Point-in-Time Copy functions are used for data duplication, and the Remote Mirror and Copy functions are used for data migration and disaster recovery.

Metro Mirror

Metro mirroring is a function of the IBM System Storage® Server. The data that is stored in independent disk pools data is on disk units located in the System Storage Server. This solution involves replication at the hardware level to a second storage server using IBM System Storage Copy Services. An independent disk pool is the basic unit of storage for the System Storage Peer-to-Peer Remote Copy (PPRC) function. PPRC provides replication of the independent disk pool to another System Storage Server.

You also have the ability to combine this solution with other System Storage-based copy services functions, including IBM FlashCopy®, for save window reduction. Metro Mirror data transfer is done synchronously. You must also be aware of the distance limitations and bandwidth requirements associated with transmission times as with any solution when synchronous communications are used.

Global Mirror

Global Mirror uses the same base technology as Metro Mirror except the transmission of data is done in an asynchronous manner and FlashCopy to a third set of disks is required to maintain data consistency. Because this data transmission is asynchronous, there is no limit to how geographically dispersed the System Storage servers can be from each other.

Metro Global Mirror

Metro Global Mirror is a method of continuous, remote data replication that operates between three sites that varying distances apart. Metro Global Mirror combines Metro Mirror synchronous copy and Global Mirror asynchronous copy into a single session, where the Metro Mirror target is the Global Mirror source. Using Metro Global Mirror and Metro Global Mirror with HyperSwap, your data exists on a second site that is less than 300 KM away, and a third site that is more than 300 KM away. Metro Global Mirror uses both Metro Mirror and Global Mirror Failover / Failback to switch the direction of the data flow. This ability enables you to run your business from the secondary or tertiary sites.

3.3.2 Site replication options

As data is key for any business continuity, utmost importance is provided to keep the data protected from planned or unplanned site failures. Accordingly, each organization has its own set of rules or requirements for safeguarding data. Depending on the organization's requirements, these are some of the most common storage replication options available:

- ▶ Two-site storage replication
- ▶ Three-site storage replication
- ▶ Four-site storage replication

In this section, we describe synchronous and asynchronous site replication options.

Synchronous: Nearline storage and two-site storage replication

Near-online, or nearline, storage describes storage that is a compromise between online storage, which supports very rapid access to data, and offline storage, which supports backups and infrequent access to data.

A two-site solution exists when synchronous replications happen between the primary and secondary or local nearline disks located at the primary site using storage based Metro Mirror (PPRC) technology (see Figure 3-3). This replication is application independent and may have a performance impact on the application. This replication can also be used for other activities such as data migration between devices and workload migration to an alternate site. The Recovery Point Objective (RPO) with this option would be equivalent to zero, as the data is always synchronized.

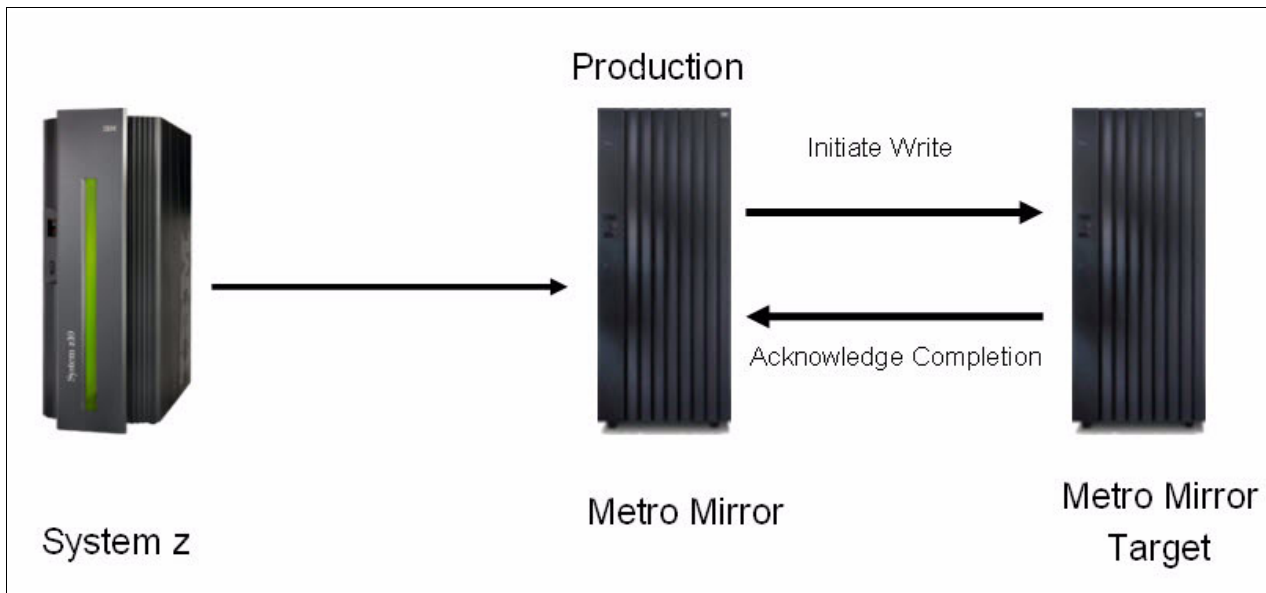


Figure 3-3 Two-site synchronous mirroring using Metro Mirror

Two-site synchronous mirroring happens in the following four steps:

1. The server sends a write request to the primary or production storage server.
2. The primary storage server initiates the secondary write operations to the target secondary storage server.
3. After the write completes in the secondary server, the secondary storage server notifies the primary storage server with a write acknowledgement.
4. When the acknowledgement is received, the primary storage server acknowledges with a device I/O end notification.

Metro Mirror is recommended when the following conditions are true:

- ▶ The recovery storage system has the same (or similar) types and features as the primary application system.
- ▶ You can accept some performance impact to application write I/O operations at the primary location (because recovery is on a disk-by-disk basis).
- ▶ The distance between the primary and secondary storage systems is under 300 km (approximately 186 miles).

Asynchronous: Two-site replication

An asynchronous remote copy approach is usually required when the distance between the local site and the remote site is beyond an efficient distance for a synchronous remote copy solution.

In an asynchronous data replication environment, an application's write I/O operation goes through the following steps:

1. Write the application data to the source storage disk subsystem cache.
2. Present a successful I/O operation completion message to the host server. The application can then immediately schedule the next I/O operation.
3. Replicate the data from the source storage disk subsystem cache to the target storage disk subsystem cache.
4. Send an acknowledgement to the source storage disk subsystem that data successfully arrived at the target storage disk subsystem.

In asynchronous replication, the data transmission and the I/O operation completion acknowledgements are independent processes. This results in virtually no application I/O operation impact, or at most, to a minimal degree only. This is a good solution for when you need to replicate over long distances.

Asynchronous transfer of an application's primary storage writes to secondary storage and allows mirroring over long distances with minimal impact to host performance.

Three-site replication

Three-site replication combines a typical three-site disaster recovery (DR) solution with an asynchronous disk based solution. The typical three-site DR solution replicates from the primary site to the DR site for applications running at the primary site. Then, the asynchronous disk based replication solution would use Global Copy (PPRC-XD) or Global Mirror to replicate from the DR site to the primary site for applications running at the DR site.

The three-site replication solution from the primary site to the DR site would ensure close to zero data loss. Each production disk volume would have a corresponding near site disk volume to which it would do a synchronous replication. This target volume would act as the source volume for an asynchronous replication to the remote site.

The replication solution for applications at the DR site is an asynchronous disk based replication to the primary site. The applications residing at the DR site are non-critical batch applications and hence can get by with a little data loss due to asynchronous replication.

Three-site replication is shown in Figure 3-4.

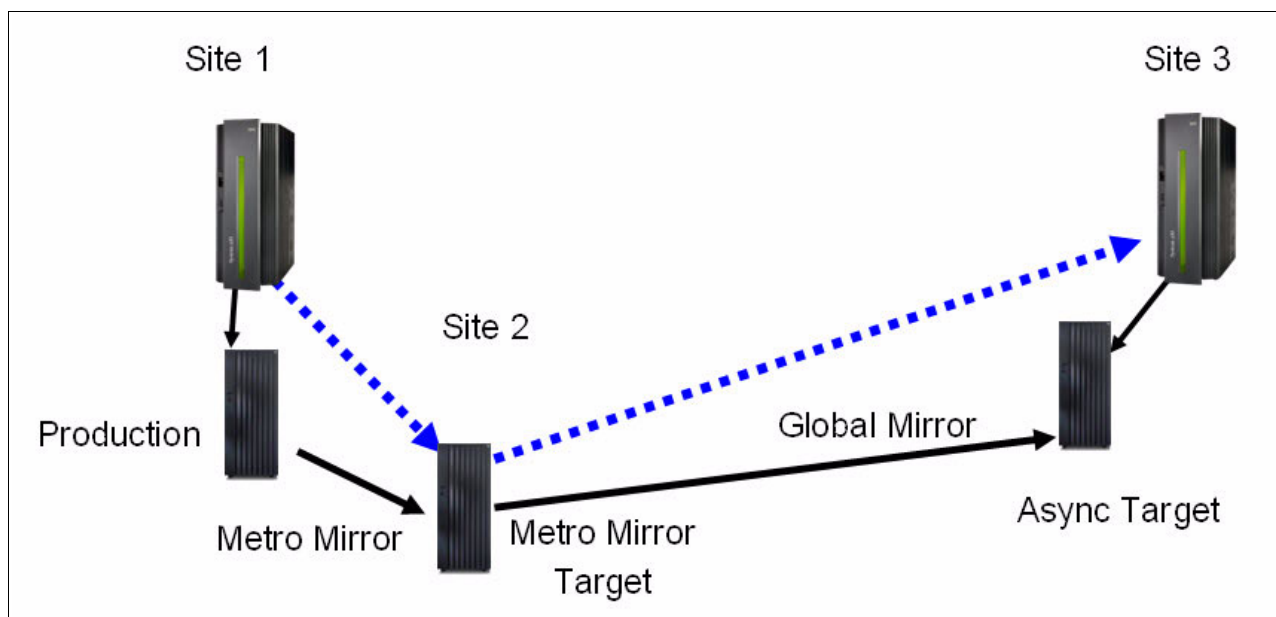


Figure 3-4 Three-site configuration

Four-site replication

In the four-site replication option, the setup would be a combination of two three-site DR solutions functioning back to back for two different sets of volumes (in opposite directions). Each three-site pair would function similar to a three-site DR solution, ensuring close to zero data loss. Each production disk volume would have a corresponding near site disk volume to which it would do a synchronous replication. This target volume would act as the source volume for an asynchronous replication to the far-off site. The four-site replication configuration is shown in Figure 3-5.

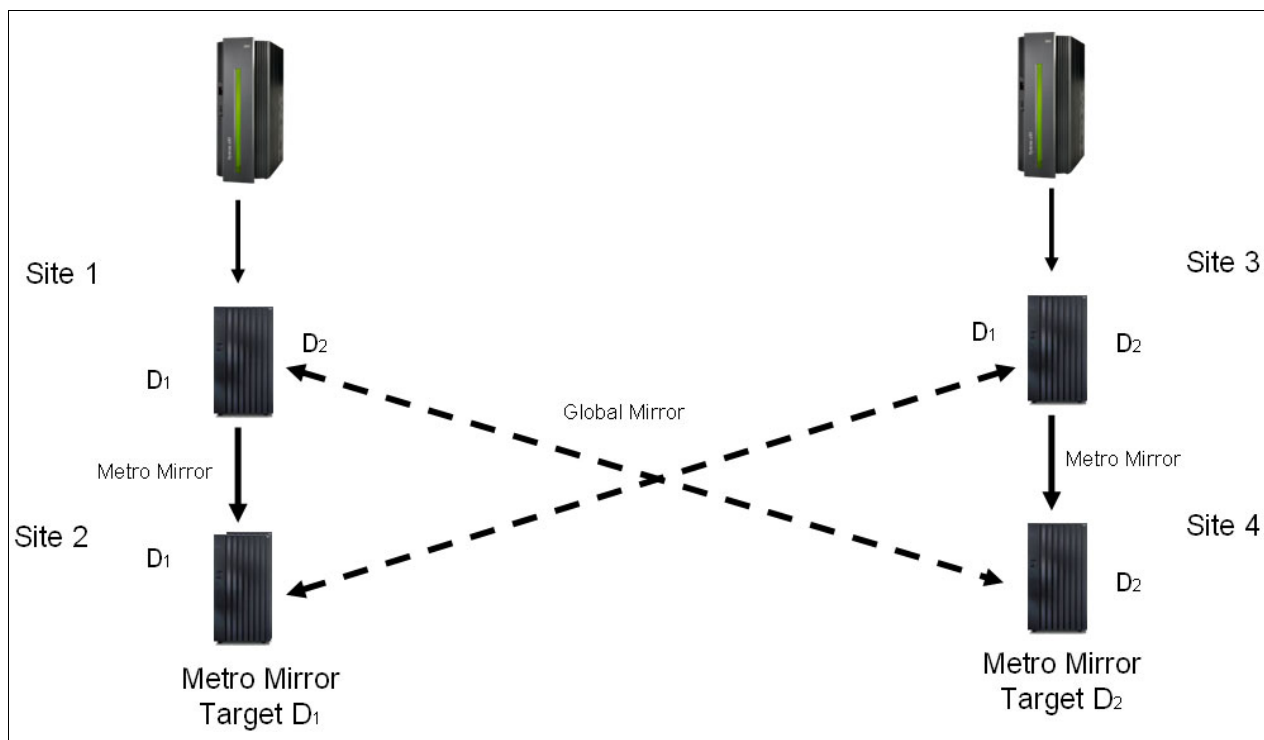


Figure 3-5 four-site configuration - two three-site configurations with multiple redundancy

The following items are illustrated in Figure 3-5:

- D1 are the volumes residing on the primary Site.
- D1' is the PPRC volume for D1 residing on the nearline disk.
- D1'' is the PPRC-XD asynchronously copied volume for D1' residing in the disk at the DR site.

The same situation applies to D2 volumes as well. From D1 to D1' and D2 to D2', the volumes are being synchronously copied. Hence D1' and D2' are up to date with their Source volumes. In case the disks corresponding to D1 or D2 fail, operations can continue using the respective PPRC Target volume disks.

3.4 Operating system high availability

This section describes some of the features related to high availability that can be found in the operating systems, z/VM and Linux on System z.

3.4.1 Operating system storage high availability features

In 3.3, “Storage high availability” on page 34, we described the high-availability capabilities provided by the storage servers, such as the IBM DS8000 family. These features are designed to ensure that the storage infrastructure remains available at all times. However, making sure that the operating systems and applications that make use of the storage space have access to the storage at all times requires additional configuration at the OS level. Different types of configurations apply depending on the disk technology chosen.

Direct Access Storage Devices (DASD)

DASD, also known as Extended Count Key Data (ECKD) devices, or even 3390 devices, are the traditional storage devices for System z operating systems. As such, these devices benefit from the highly-available design of the mainframe’s I/O subsystem.

Most of the configuration takes place in the I/O Configuration DataSet (IOCDs). In an extract of the IOCDs definition shown in Example 3-1, we can see that Control Unit Number (CNTLUNIT) 9900 can be reached by 4 logical paths, 40 through 43. This control unit controls access to devices in the 9900 range.

Example 3-1 Extract of ITSOZVM I/O definition

```
CNTLUNIT CUNUMBR=9900, *
    PATH=((CSS(0),40,41,42,43),(CSS(1),40,41,42,43),(CSS(2),
    40,41,42,43),(CSS(3),40,41,42,43)),UNITADD=((00,256)), *
    LINK=((CSS(0),1D,1D,5D,59),(CSS(1),1D,1D,5D,59),(CSS(2),
    1D,1D,5D,59),(CSS(3),1D,1D,5D,59)),CUADD=2,UNIT=2107
IODEVICE ADDRESS=(9900,048),CUNUMBR=(9900),STADET=Y,UNIT=3390B
IODEVICE ADDRESS=(9900,128),UNITADD=80,CUNUMBR=(9900), *
    STADET=Y,SCHSET=1,UNIT=3390A
IODEVICE ADDRESS=(9970,002),CUNUMBR=(9900),STADET=Y,UNIT=3390B
IODEVICE ADDRESS=(997F,001),CUNUMBR=(9900),STADET=Y,UNIT=3390B
```

This configuration is confirmed when querying the z/VM setup, as shown in Example 3-2.

Example 3-2 z/VM disk I/O configuration query

```
q path to 9920
Device 9920, Status ONLINE
  CHPIDs to Device 9920 (PIM) : 40 41 42 43
    Physically Available (PAM) : + + + +
    Online (LPM) : + + + +
    Transport Mode supported : + + + +
    Legend      + Yes - No
Ready; T=0.01/0.01 09:48:37
q chpid 42
Path 42 online to devices 9910 9911 9912 9913 9914 9915 9916 9917
Path 42 online to devices 9918 9919 991A 991B 991C 991D 991E 991F
Path 42 online to devices 9920 9921 9922 9923 9B2B 9B2C 9B2D 9B2E
Path 42 online to devices 9B2F
Path 42 offline to devices 9900
Ready; T=0.01/0.01 09:48:41
```

As soon as there is more than one path available to reach a device, the I/O subsystem will handle any path failures. z/VM gets notified of errors and will try to reroute the I/O requests to the remaining paths. When used as dedicated devices in Linux, it is the DASD driver's responsibility to transparently handle any I/O path failures.

Small Computer System Interface (SCSI) devices

Using SCSI devices, sometimes referred to as FCP devices, in a Linux on System z environment is common, especially for new customers with no z/OS skills or background.

SCSI devices are usually dedicated to the Linux virtual machine, such as z/VM, being nothing but a pass-through with no involvement in the I/O operation. SCSI I/O operations are less integrated with the mainframe I/O subsystem, therefore some additional configuration is required to ensure a proper level of availability at the Linux level. More than one path needs to be available and configured in the SAN infrastructure, as well as on System z. The use of these multiple paths to access LUN devices then needs to be configured in Linux.

Note: Covering all the specifics of the configuration and use of SCSI devices with z/VM and Linux is far beyond the scope of this book. For more information, see *How to use FC-attached SCSI devices with Linux on System z*, SC33-8413-07 at this website:

http://www-01.ibm.com/support/knowledgecenter/linuxonibm/liaaf/lnz_r_ts.html

Multipathing

Multipathing means that there is more than one physical path between the CPU and its mass storage devices through buses, controllers, switches, and bridge devices connecting them.

A simple example would be a SCSI disk connected to two SCSI controllers on the same computer or a disk connected to two Fibre Channel ports. Should one controller, port or switch fail, the operating system can route I/O through the remaining controller transparently to the application, with no changes visible to the applications, other than perhaps incremental latency

Linux makes use of a multipathing driver, **dm_multipath**, to configure several access paths to one given logical unit (LUN), as shown in Figure 3-6 on page 41. Linux multipathing provides I/O failover and path load sharing for multipathed block devices.

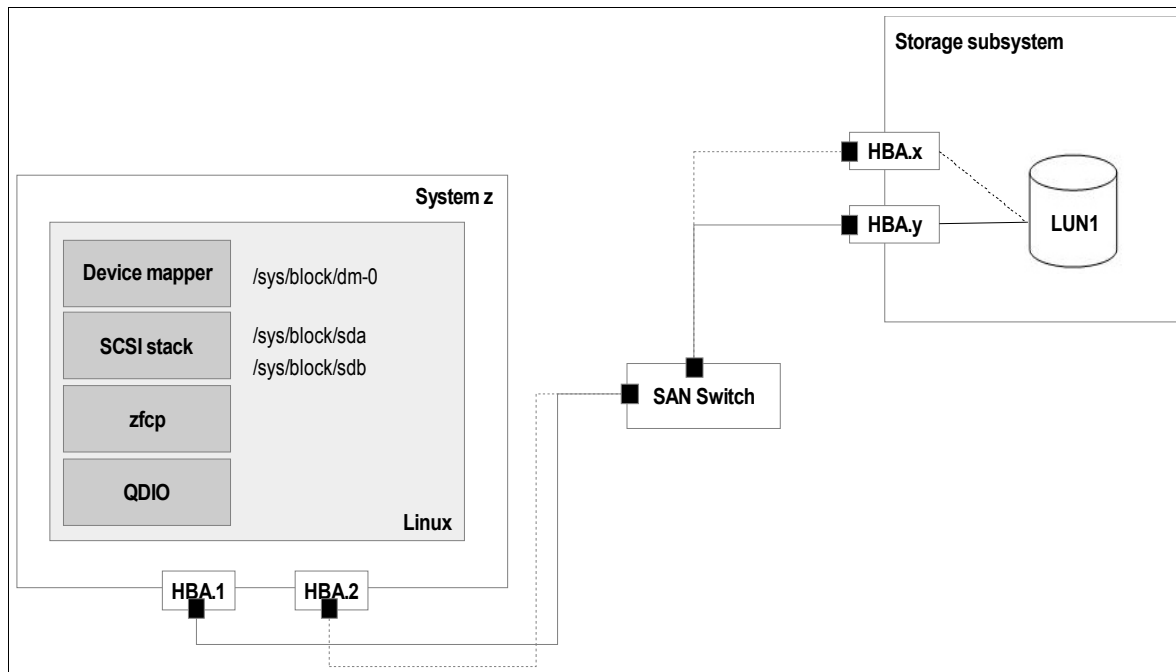


Figure 3-6 Linux multipathing driver

The configuration file for multipath, `/etc/multipath.conf`, can be edited to specify the desired grouping policy. Two commonly used policies are used to group paths together:

- Failover

The failover policy keeps the volume accessible in case of a single failure in the connecting hardware. If one path fails, the operating system will reroute the I/O requests through one of the remaining paths, with no visible changes to the applications.

- Multibus

The multibus policy uses all available paths in a priority group in a round-robin manner, switching paths after a configurable number of I/O requests (1000 by default, which can be overridden using the `rr_min_io` parameter). Choosing the multibus policy over the failover policy can help overcome the throughput limits of a single path.

Note: Multipathing tools can be used with SCSI devices with all Linux distributions, and when using ECKD devices with static PAV configuration in older Linux distributions (prior to Red Hat Enterprise Linux 6 and SUSE Linux Enterprise Server 11). Note also that HyperPAV has made multipath tools unnecessary.

3.4.2 z/VM high availability features

z/VM is a System z operating system. As such, it is developed to provide the highest quality of service and availability to the virtual machines it hosts.

z/VM includes many availability-related features. For instance, CP, the hypervisor component in a z/VM system, includes transparent error handling and recovery, whenever possible. There are several built-in transparent network failover techniques available, when using dedicated OSA cards or virtual switches for instance, as explained in 3.6, “Network high availability” on page 52.

These techniques try to guarantee the availability of a z/VM Logical Partition (LPAR). However, they do not prevent unrecoverable hardware errors that could bring the system down.

z/VM version 6 Release 2 introduced two new features that can be used to improve the availability of virtual machines in case of a hardware maintenance or failure:

- ▶ Single System Image (SSI)
- ▶ Live Guest Relocation (LGR)

Single System Image

Single System Image (SSI) is the z/VM hypervisor clustering feature. It allows you to create a z/VM hypervisor cluster of up to 4 members, which can be located on up to 4 physical machines, as described in Figure 3-7. z/VM SSI has been specifically designed to address planned outages, such as z/VM or hardware maintenance.

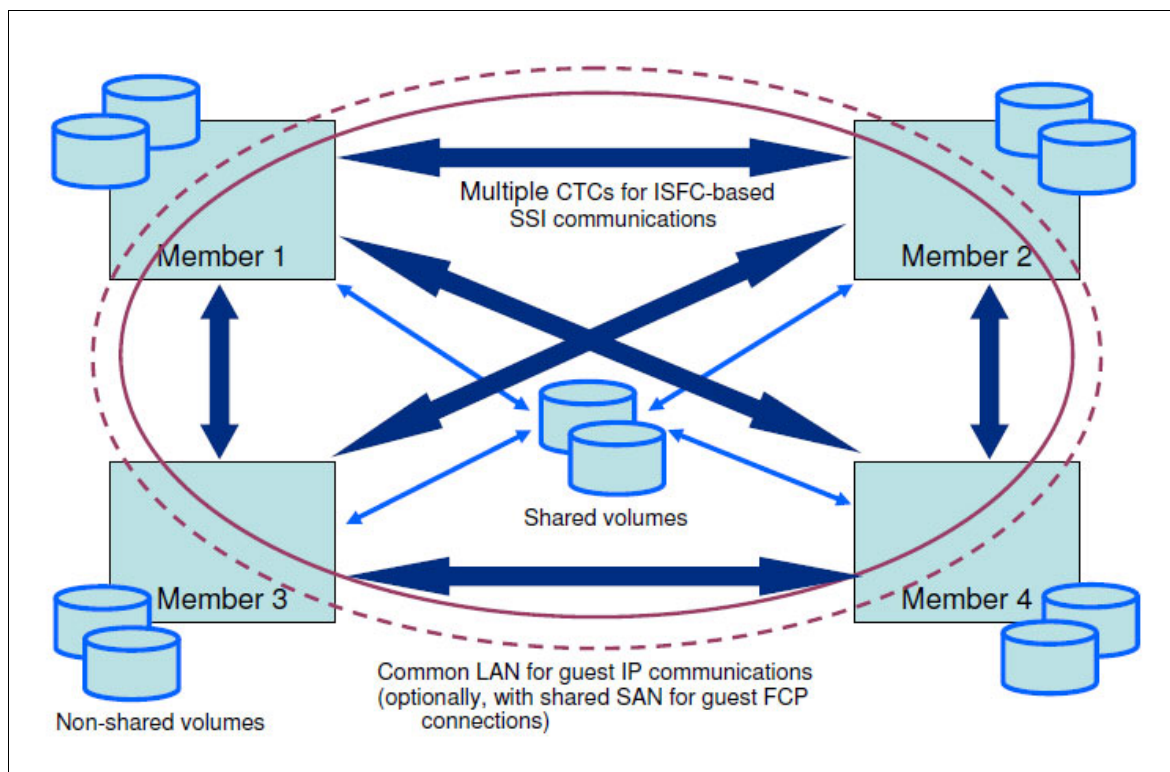


Figure 3-7 z/VM Single System Image

z/VM SSI uses a set of shared 3390 devices to host the configuration files that are common to all members of the cluster, including these:

- ▶ SYSTEM CONFIG
- ▶ USER DIRECTORY

The z/VM SSI feature is not designed to cope with unplanned outages by itself. However, it will help system administrators to reduce downtime in the unlikely event of a z/VM LPAR crash. All virtual machines definitions and resources being available in the remaining members of a cluster, the system programmers team will only have to restart the required machines in one of the remaining members of the cluster.

Live guest relocation

Live guest relocation, or LGR, is a z/VM feature new in version 6.2 that allows a Linux virtual machine to be dynamically moved from one LPAR to another LPAR in the same cluster, in a transparent manner for the applications.

LGR itself does not bring anything in terms of high-availability for the z/VM environment. It improves availability of the Linux machines running in a z/VM LPAR, or a System z, that needs servicing for instance. The Linux machines can be dynamically relocated into one of the remaining members of the cluster during the maintenance operations, hence showing no downtime from the application's perspective.

Dynamic reconfiguration

z/VM has the capability to change its configuration whenever is needed, after it is properly planned and configured. This capability can prove itself useful, especially when running an active-passive configuration, as described in 2.5.2, "Active / passive configuration".

For instance, the passive z/VM LPAR can be defined with only one active processor (so that the hypervisor can be started) while being passive, with additional capacity (memory or processors) being brought online when the partition becomes active, to sustain the increased load. Example 3-3 illustrates how memory can be added on the fly to a running z/VM LPAR.

Example 3-3 Dynamic memory reconfiguration

```
q sto
STORAGE = 32G CONFIGURED = 32G INC = 256M STANDBY = 2G RESERVED = 0
Ready(00003); T=0.01/0.01 10:21:52
set storage +2G
STORAGE = 34G CONFIGURED = 34G INC = 256M STANDBY = 0 RESERVED = 0
```

The same principle could be applied to vary processors online after adding them to the LPAR.

This allows for keeping resources allocated to the standby LPARs at their minimum. The additional resources being only required for the time of the outage, they will be brought back online on demand, at the system programmer's request, and stay available as long as necessary. After the situation has been brought back to normal, the extra resources can be deallocated from the standby systems.

Geographically Dispersed Parallel Sysplex (GDPS)

When GDPS is in place at a customer site to provide high-availability and disaster recovery capabilities to a z/OS environment, it is also possible to include z/VM and Linux on a System z environment under the same mechanisms. See 1.2.3, "HA architecture terminology" for more details about GDPS.

3.4.3 Linux high availability features

When it comes to implementing clusters on Linux (on System z), there are basically four options:

- ▶ Linux-HA
- ▶ SUSE Linux Enterprise High Availability Extension
- ▶ Red Hat Enterprise Linux Add-on
- ▶ IBM Tivoli System Automation

The following section describes these options and their differences.

Linux-HA

The Linux-HA project is hosted at the following website:

<http://www.linux-ha.org>

It maintains a set of building blocks for high availability clusters, including these:

- ▶ A cluster messaging layer, named Heartbeat:

Heartbeat provides the clustering capabilities (messaging and membership) that ensure high availability of critical resources such as data, applications, and services. It provides monitoring, failover, and fallback capabilities to Heartbeat-defined resources.

- ▶ A large number of resource agents for a variety of applications, named resource-agents:

A resource agent is a standardized interface for managing a cluster resource. It translates a standard sets of operations (for instance start, stop, or status) into steps specific to the application or resource.

Three types of resource agents co-exist in Linux-HA:

- Linux standard base (LSB) resource agents, the initialization scripts shipped by the distribution, as found in /etc/init.d
- Open Clustering Framework (OCF) resource agents that follow the OCF specification, an extension of the LSB specification
- Heartbeat resource agents that are the legacy scripts shipped with the previous versions of the Heartbeat software

The Linux-HA resources-agents project provides resources agents for various middleware applications such as Oracle, IBM DB2, IBM WebSphere Application Server, SAP, and Apache.

- ▶ Cluster Glue, which is a “plumbing” library and error reporting toolkit:

Cluster Glue is a set of utility libraries and tools that are used in a clustered environment. It includes several components, such as:

- A Local Resource Manager (LRM), which acts as the interface between the resource agents and the Cluster Resource Manager (CRM), querying the state of a resource and reporting it back to the CRM.
- STONITH (Shoot The Other Node In The Head), which is a mechanism for node fencing. This mechanism is used to evict a non-responding member from the cluster configuration.
- hb_report, an advanced error reporting tool.
- The Cluster Plumbing Library, a low-level library for intra-cluster communications.

- ▶ Pacemaker, a cluster resource manager (CRM), maintained separately by ClusterLabs:

<http://www.clusterlabs.org>

It achieves maximum availability for the cluster resources by detecting and recovering from node and resource-level failures by making use of the messaging and membership capabilities provided by the Heartbeat infrastructure.

The tools provided by the Linux-HA project are multi platform, which means they can be run on Linux running on any kind of platform. However, there are no Linux-HA packages available in the standard Linux on System z distribution medias. They have to be hand-compiled to be used on Linux on System z.

Note: For more details about using Linux-HA on System z, see *Achieving High Availability on Linux for System z with Linux-HA Release 2*, SG24-7711.

SUSE Linux Enterprise High Availability Extension

The SUSE Linux Enterprise High Availability Extension (SLE HAE) is provided free of charge to customers running SUSE Linux Enterprise Server version 11 for System z.

This extension, available on all SUSE supported platforms (Intel 32 and 64bits, Itanium, Power, and System z), provides the packaged tools to create clusters. SLE-HAE includes the following components:

- ▶ Corosync, which is part of the OpenAIS project and is licensed under the new BSD License, is used as the messaging layer.
- ▶ Pacemaker, which is a scalable, high availability cluster resource manager, which was discussed in the previous section under the heading, “Linux-HA”.
- ▶ OCF resource agents to manage resources such Apache webserver, IPv4 or IPv6 addresses, IBM WebSphere Application Server and others.
- ▶ Integrated management tools:
 - Yet another Setup Tool (YaST) modules
 - Pacemaker GUI

SLE-HAE also ships with additional components to create highly available filesystems and logical volumes, using these products:

- ▶ Oracle Cluster File System 2 (OCFS2) and Clustered Logical Volume Manager (cLVM)
OCFS2 allows concurrent read and write access on filesystem from different hosts, while cLVM permits to aggregate several volumes into a logical larger one to be shared in a cluster.
- ▶ Distributed Replicated Block Device (DRBD) the technology used to provide disk replication at the Linux operating system level.

Red Hat Enterprise Linux High Availability Add-on

For a very long time, Red Hat has provided high availability functions using dedicated packaging known as the Red Hat Enterprise High Availability Add-on. As of Red Hat Enterprise Linux 6, the High Availability Add-on included these features:

- ▶ Corosync/OpenAIS as the messaging layer
- ▶ Resource Group (Service) Manager, or RGManager, runs under the cluster manager, or CMAN, as the Cluster Resource Manager
- ▶ Fence and resource agents
- ▶ Configuration tools such as these:
 - The command-line tool, Cluster Configuration System (CCS)
 - A web interface named **luci**

Note: As of Red Hat Enterprise Linux 7, RGManager has been replaced with Pacemaker.

High-availability and clustering functions for filesystems and logical volumes are provided by the following technologies:

- ▶ Global File System 2 (GFS 2)
- ▶ Clustered Logical Volume Manager (CLVM)

The Red Hat Enterprise Linux High Availability Add-on is currently not supported by Red Hat on System z. However, it has been ported to System z by an IBM Business Partner, Sine Nomine, which also provides support for this add-on. For more information about this product, see this website:

<http://www.sinenomine.net/products/linux/hao>

3.5 Software solutions for high availability

Operating systems have features and functions that can be employed to improve the availability of their systems and applications. In addition to the operating system high availability features, additional HA software products can be used to enhance and increase the number of HA options. Some of these HA software products and their capabilities are described in this section.

IBM Tivoli System Automation

The IBM Tivoli System Automation family of products help reduce the frequency and duration of service disruptions with advanced policy-based automation to enable the high availability of critical applications and middleware running on a range of hardware platforms and operating systems. The product portfolio also provides valuable operational capabilities to help IT staff efficiently and effectively manage the availability and performance of the IT services they deliver, decreasing costs and reducing risks. One of the key management capabilities that the product family offers is a single point of control for managing heterogeneous high availability solutions and IT service dependencies that can span Linux, AIX, Windows, Solaris, and IBM z/OS.

The System Automation family includes the following products:

- ▶ IBM Tivoli System Automation Application Manager (TSA AM)

TSA AM offers a single point of control to manage heterogeneous business application landscape. It helps coordinate and manage application availability across cluster technologies, so you can better control your enterprise business services. TSA AM can be used to establish one operations and automation team that is responsible for Linux, AIX, Windows, and Solaris applications. SA Application Manager can simplify problem determination and resolution as well.

IBM Tivoli SA AM can be used to facilitate High Availability and Disaster Recovery management of composite business applications (Cross-Cluster, Cross-System, and Cross-Site). It automates startup, shutdown, and re-synchronization of composite business applications running on different platforms. It integrates these components:

- ITM/ ITCAM managed Endpoints for a consistent management of End-to-End composite business applications
- Various High-Availability Clusters (for example, SA MP, SA z/OS, PowerHA, MSCS, Veritas Cluster Server)
- Application components with replicated storage devices in order to provide a multi-site disaster recovery solution

The focus of System Automation is to automate the availability of IT resources. This availability is accomplished by starting and stopping resources automatically and in the correct sequence. Resources are located on a system, which is referred to as *node* in the context of a cluster. Resources controlled by System Automation can be applications, services, mounted disks, network addresses or even data replication. Basically anything on a node which can be monitored, started and stopped with help of commands or through an API. For each resource, Tivoli System Automation provides an availability state and offers a way to start and stop them.

► IBM Tivoli System Automation for Multiplatforms (SA MP)

SA MP offers high availability and policy-based automation for applications and services across heterogeneous environments. It reduces frequency and duration of service disruptions for critical applications and middleware running on heterogeneous platforms and virtualization technologies.

Figure 3-8 provides a graphical overview of SA MP.

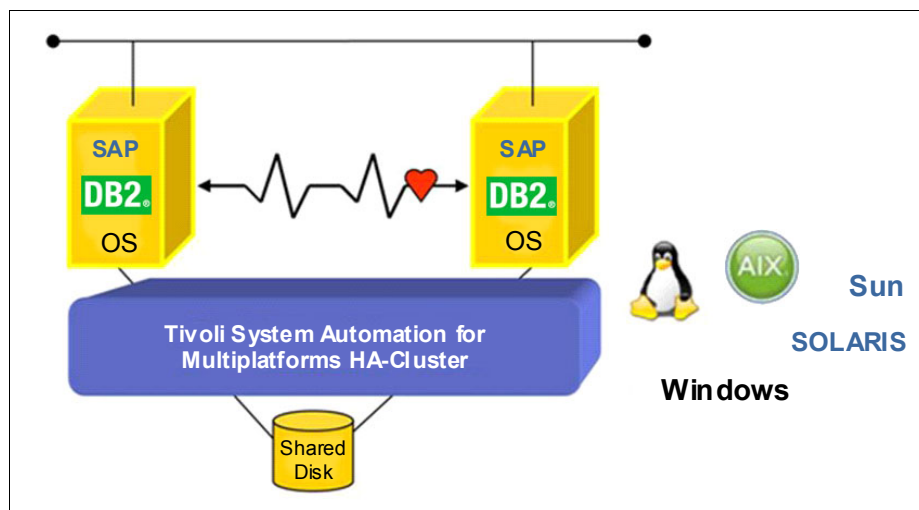


Figure 3-8 IBM Tivoli SA MP overview

SA MP manages availability of business applications, running in Windows, Linux, Solaris and AIX systems. It provides a so called high-availability cluster. Customers can make use of a set of sample policies to create true high availability solutions - with the freedom to create own automation policies for all application landscapes they want to keep highly available.

IBM Tivoli System Automation Multiplatform provides:

- Fast detection of outages through active monitoring and heartbeats
- Sophisticated knowledge about application components and their relationships
- Quick and consistent recovery of failed resources and whole applications either in place or on another system of a cluster

SA MP can help building a cluster as it comes with its own cluster infrastructure. Linux systems or clusters in System z LPARs or under z/VM are supported.

If you use System Automation for Multiplatforms, the following advantages and improvements apply:

- Application availability is improved by reducing the number and duration of incidents that impact availability.
- Policy-based automation reduces skill requirements and improves automation quality and flexibility.
- Plug and play automation modules can reduce automation costs, implementation time and support effort.
- Less operator errors and effort through operations at the application level using a Web-based GUI.

Using IBM Tivoli SAMP, you can implement powerful, policy-based automation with start, stop, and location dependencies. It provides consistent automation across all supported platforms. SAMP can help protect critical resources through quorum. It offers the ability to coordinate systems through a dead man switch, and a disk and network tiebreaker. You can build small, large, and complex clusters which reduces the number of required backup systems.

IBM Tivoli SAMP's goal driven automation avoids operator errors. It integrates with Tivoli System Automation Application Manager to automate and operate composite applications and to facilitate reporting. Both products have web-based, easy to use, and powerful graphical user interfaces. System Automation for Multiplatforms helps building a cluster as it comes with its own cluster infrastructure. Linux systems or clusters in System z LPARs or under z/VM are supported.

► IBM Tivoli System Automation for z/OS

IBM Tivoli System Automation for z/OS is a policy-based, self-healing, high-availability solution to maximize efficiency and availability of critical systems and applications on System z (z/OS) as shown in Figure 3-9. IBM Tivoli System Automation for z/OS provides the following three major functions:

- System Automation provides plug and play automation modules for CICS, IMS, DB2, SAP, WebSphere and Geographically Dispersed Parallel Sysplex™ (GDPS), the leading disaster recovery solution for System z.
- I/O operations helps to increase system availability and operator productivity. Operators can make operational changes to IBM System z I/O configurations FICON, ESCON and non-ESCON in a safe, system-integrated way.
- Processor operations enables remote console access and external automation during IML, IPL and problem determination of System z. Highlights include LPAR Capacity Management and support for Linux as a z/VM guest.

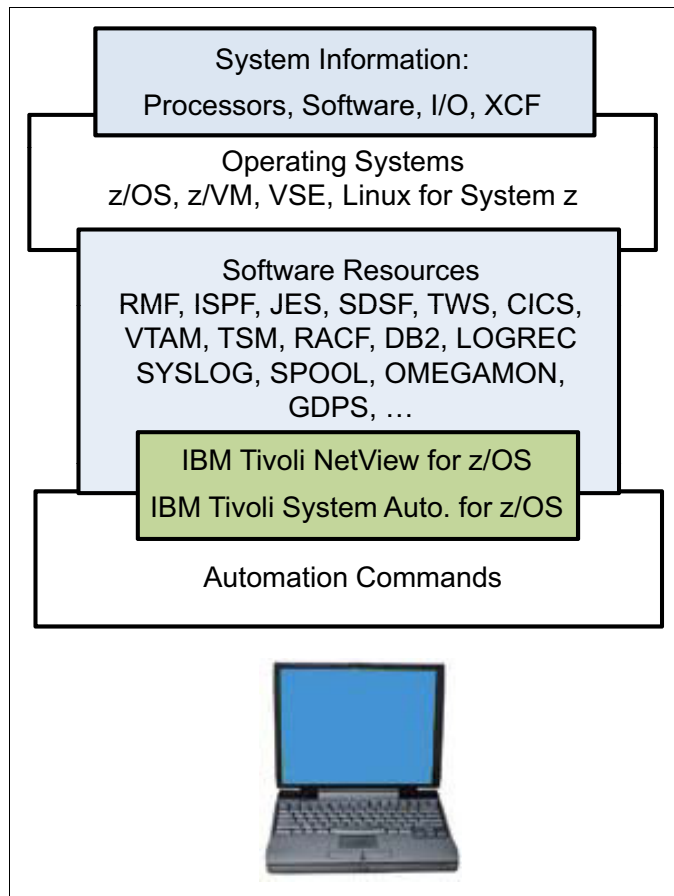


Figure 3-9 Monitor, control and automation functions

System operations monitors and controls operating system components and middleware such as CICS®, IMS™, and DB2®. With system operations, you can automate Parallel Sysplex applications. SA z/OS can automate applications distributed over a sysplex by virtually removing system boundaries for automation through its automation manager/automation agent design. SA z/OS reduces the complexity of managing a Parallel Sysplex through its goal driven automation and its concepts, such as grouping and powerful dependency support, which enable you to model your configuration. Single systems are also fully supported; the automation scope is then just one system.

Processor operations monitors and controls processor hardware operations. It provides a connection from a focal point processor to a target processor. With NetView on the focal point system, processor operations automates operator and system consoles for monitoring and recovering target processors. Processor operations allows you to power on and off multiple target processors and reset them, perform IPLs, set the time of day clocks, respond to messages, monitor status, and detect and resolve wait states.

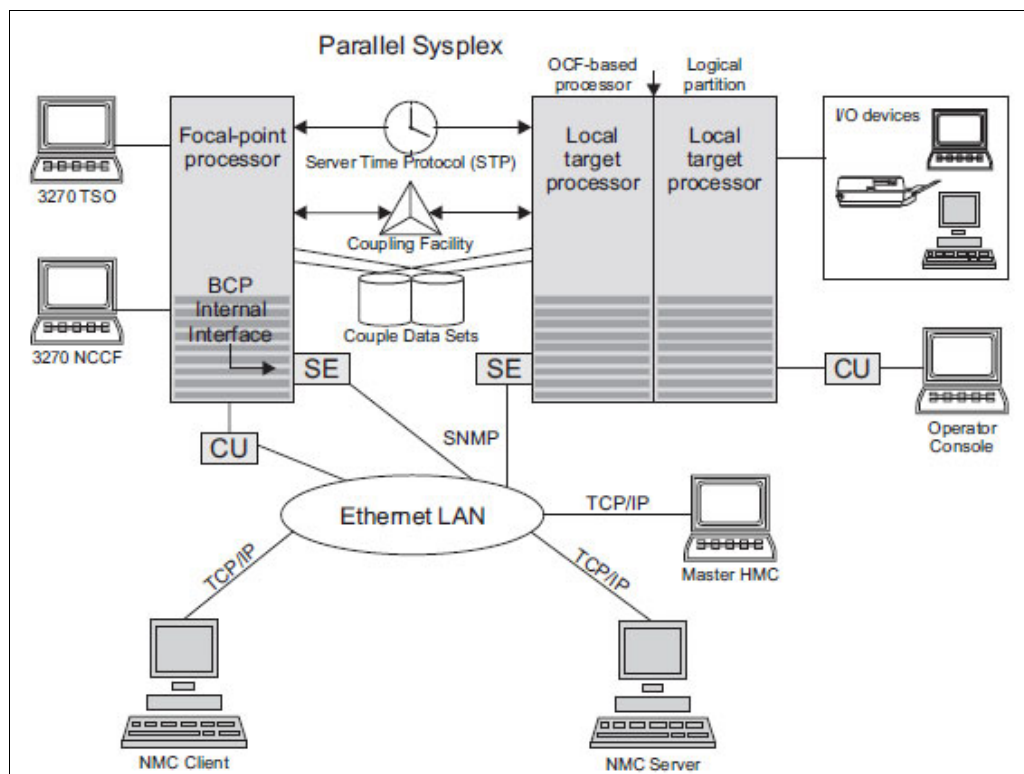


Figure 3-10 Tivoli System Automation for z/OS configuration

I/O operations provides a single point of control for managing connectivity in your active I/O configurations, (Figure 3-10). It takes an active role in detecting unusual I/O conditions and lets you view and change paths between a processor and an input/output device, which can involve using dynamic switching: the enterprise systems connection (ESCON®) or fiber channel connection (FICON®) switch. I/O operations changes paths by letting you control channels, ports, switches, control units, and input/output devices. You can do this through an operator console or API.

3.5.1 Linux with z/OS high availability solutions

When Linux and z/OS are present in the IT environment (such as that shown in Figure 3-11), there are additional software offerings to extend high availability options for Linux servers and especially end-to-end applications. Complex applications that pass through multiple routers, firewalls, distributed servers and mainframe z/OS systems are difficult environments for ensuring high availability. IBM's solution for creating highly available end-to-end (z/OS and Linux) environments is the Geographically Dispersed Parallel Sysplex (GDPS).

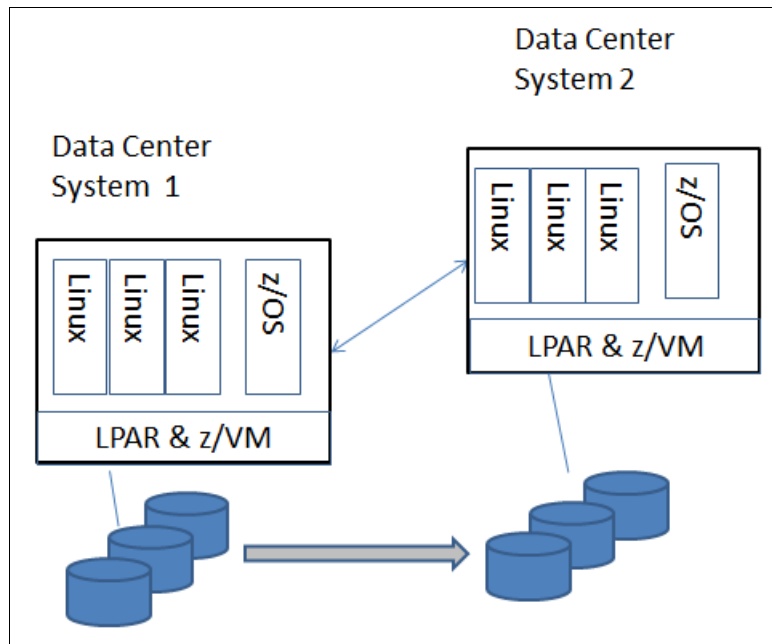


Figure 3-11 Data centers with Linux and z/OS

GDPS is actually a collection of several offerings, each addressing a different set of high availability goals, that can be tailored to meet the HA targets for your business. Each offering leverages a combination of server and storage hardware or software-based replication and automation and clustering software technologies, many of which are described in more detail later in this chapter.

In addition to the infrastructure that makes up a given GDPS solution, IBM also includes services, particularly for the first installation of GDPS and optionally for subsequent installations, to ensure the solution meets and fulfills your business objectives.

The following list provides brief descriptions of each offering, with the HA objectives it is intended to address. Your IT data center locations and HA requirements will influence your decision on which option best fits your business needs.

GDPS/PPRC	This provides near-continuous availability solution across two sites separated by metropolitan distances. The solution is based on the IBM PPRC synchronous disk mirroring technology.
GDPS/PPRC HyperSwap Manager	This is a near-continuous availability solution, but for a single site. The solution is based on the same technology as GDPS/PPRC, but does not include much of the systems automation capability that makes GDPS/PPRC a more complete DR solution.
GDPS Metro/Global Mirror	As either a 3-site or a symmetrical 4-site configuration is supported: GDPS Metro/Global Mirror 3-site is a three-site solution that provides high availability across two sites within metropolitan distances and disaster recovery to a third site, in a different region, at virtually unlimited distances. It is based on a cascading mirroring technology that combines PPRC and Global Mirror. GDPS Metro/Global Mirror 4-site is a symmetrical four-site solution that is similar to the 3-site solution in that it provides high availability within region and disaster recovery cross region. In addition, in the 4-site solution, the two regions are configured symmetrical such that the same levels of HA and DR protection is provided, no matter which region production runs in.
GDPS Metro/z/OS Global Mirror	Provides a three-site solution that provides high availability across two sites within metropolitan distances and disaster recovery to a third site at virtually unlimited distances. It is based on a multitarget mirroring technology that combines PPRC and XRC (also known as z/OS Global Mirror on IBM storage subsystems).
GDPS/Active-Active	A multisite high availability solution at virtually unlimited distances. This solution is based on software-based asynchronous mirroring between two active production sysplexes running the same applications with the ability to process workloads in either site. End-to-end applications having their infrastructure on z/OS and Linux (Figure 3-11) can take advantage of these particular GDPS options for hosting highly available systems: <ul style="list-style-type: none"> - GDPS/PPRC, GDPS/PPRC HyperSwap Manager, and GDPS/Global Mirror with open LUN management function - GDPS/PPRC Multiplatform Resiliency for System z (also known as xDR) - GDPS/PPRC, GDPS/XRC, and GDPS/GM with Distributed Cluster Management support for Veritas Cluster Servers (VCS) - GDPS/PPRC and GDPS/GM with Distributed Cluster Management support for Tivoli System Automation Application Manager

3.6 Network high availability

System z together with the z/VM hypervisor offers a large set of networking options to choose from, illustrated in Figure 3-12:

- ▶ Dedicated Open Systems Adapter (OSA) connections
- ▶ Hipersockets
- ▶ z/VM guest LANs, among which the z/VM Virtual Switch plays a key role

This section describes the options that can be used to create highly available networking environment on System z.a

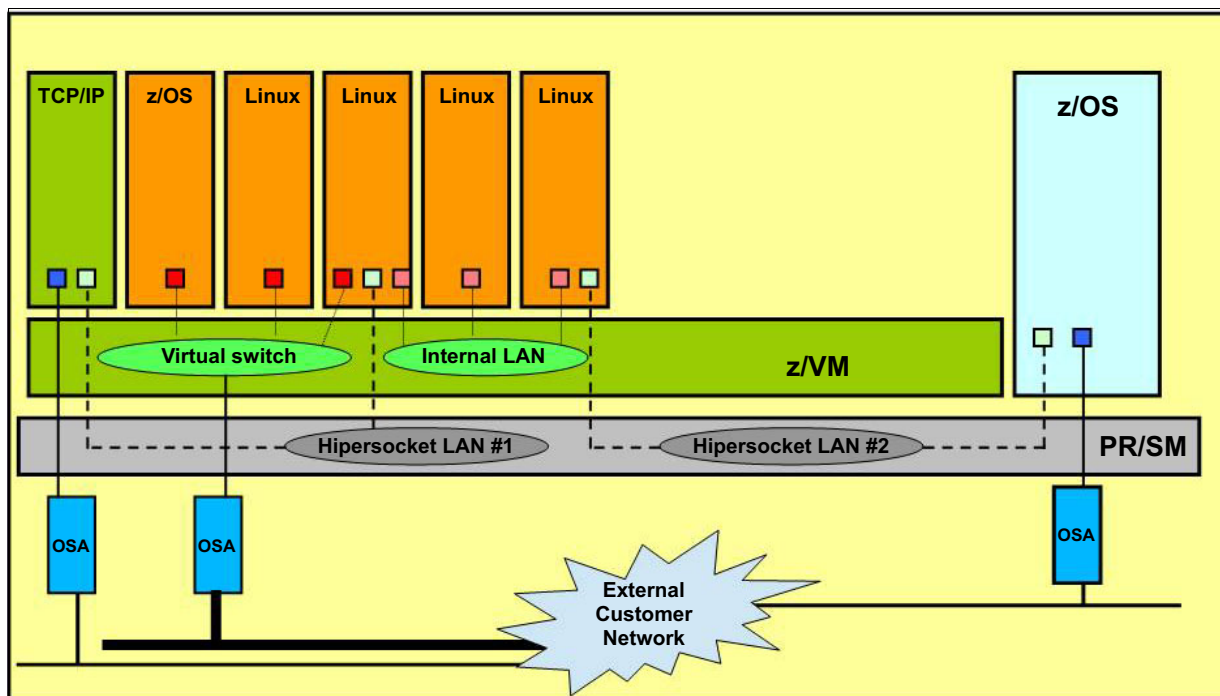


Figure 3-12 System z networking options

3.6.1 External network infrastructure

Discussing highly available external network architecture is beyond the scope of this book. However, it is important to understand that not only the System z environment, but also the external, existing, customer network environment, must be considered in an end-to-end highly available architecture. Having a highly available System z architecture is useless if a failure in the external network or architecture prevents a user from connecting to the application running on System z.

Each network vendor has its own set of technologies and recommended practices for availability. Consult your network team and network vendor for assistance.

3.6.2 Open Systems Adapter cards

Open Systems Adapter (OSA) cards are the System z specific network cards. They are used to provide external connection to the LPARs running on System z, either as dedicated devices in a machine, or connected a z/VM Virtual Switch.

Dedicated OSA connections to z/VM TCP/IP stack

Dedicating OSA devices to the z/VM TCP/IP virtual machine (the machine in charge of TCP/IP connectivity) in z/VM is a common practice. It has the benefit of isolating the z/VM administration and management network from the guests production networks. In case of a virtual switch failure or production network configuration error, the TCP/IP connection to the z/VM LPAR will still be available for investigation and correction.

Transparent fault-tolerance

The TCP/IP stack in z/VM provides transparent fault-tolerance for failed (or stopped) IPv4 or IPv6 devices, when the stack is configured with redundant connectivity into a LAN. This support is provided by the z/VM TCP/IP interface-takeover function, and applies to IPv4 queued direct I/O (QDIO) and LAN channel station (LCS) ethernet devices and to the IPv6 QDIO ethernet devices.¹

At device startup time, TCP/IP learns of redundant connectivity onto the LAN, and uses this information to select suitable backups in the case of a future failure of the device. This support makes use of address resolution protocol (ARP) flows (for IPv4 devices) or neighbor discovery flows (for IPv6 devices), so upon failure (or stop) of a device, TCP/IP immediately notifies stations on the LAN that the original IPv4 or IPv6 address is now reachable through the backup's link-layer (MAC address). Users targeting the original IP address will see no outage due to the failure, and will be unaware that any failure occurred.

Because this support is built upon ARP or neighbor discovery flows, no dynamic routing protocol in the IP layer is required to achieve this fault tolerance. To enable this support, you must configure redundancy onto the LAN. To do this:

- ▶ You need redundant LAN adapters.
- ▶ You must configure and activate multiple LINKs onto the LAN. Example 3-4 shows the definitions required in the PROFILE TCPIP configuration file.

Note: An IPv4 device cannot back up an IPv6 interface, and an IPv6 interface cannot back up an IPv4 device.

Example 3-4 Multiple LINK statements in PROFILE TCPIP provide transparent fault-tolerance.

```
[...]  
DEVICE DEV@2040  OSD 2040  NONROUTER  
LINK OSA2040 QDIOETHERNET DEV@2040  NOPATHMTU MTU 4096 IP  
DEVICE DEV@2060  OSD 2060  NONROUTER  
LINK OSA2060 QDIOETHERNET DEV@2060  NOPATHMTU MTU 4096 IP  
; (End DEVICE and LINK statements)  
HOME  
9.12.5.59 255.255.240.0 OSA2040  
9.12.7.112 255.255.240.0 OSA2060  
; (End HOME Address information)  
[...]  
START DEV@2040  
START DEV@2060  
; (End START statements)
```

To simulate the loss of one link, we issued the **DETACH 2060-2062** TCP/IP command from the TCPMAINT console. After about 30 seconds, TCP/IP automatically reassigned the IP address of the lost link to the remaining one. Example 3-5 on page 55 shows the TCP/IP console after we detached the devices 2060-2062 from the TCP/IP machine. The remaining connection, DEV@2040, gets assigned with the IP address that was previously assigned to the DEV@2060 devices.

¹ z/VM: TCP/IP Planning and Customization, SC24-6238-04

Example 3-5 Automatic, transparent IP failover in case of a link loss

```
TCPIP : 09:35:42 DTCQDI001I QDIO device DEV@2060 device number 2060:
TCPIP : 09:35:42 DTCQDI016E QDIO output queue is not operational
TCPIP : 09:35:42 DTCQDI001I QDIO device DEV@2060 device number 2060:
TCPIP : 09:35:42 DTCQDI007I Disable for QDIO data transfers
TCPIP : 09:35:42 DTCOSD082E OSD shutting down:
TCPIP : 09:35:42 DTCPRI385I Device DEV@2060:
TCPIP : 09:35:42 DTCPRI386I Type: OSD, Status: Ready
TCPIP : 09:35:42 DTCPRI387I Envelope queue size: 0
TCPIP : 09:35:42 DTCPRI497I Address: 2060 Port Number: 0
TCPIP : 09:35:47 DTCOSD246I OSD device DEV@2040: Assigned IPv4 address 9.12.7.
```

112

During our tests, when the link came back up, which means when we reattached devices 2060-2062 to TCPIP, we had to instruct TCP/IP to start using these devices again, using the following NETSTAT command:

NETSTAT OBEY START DEV@2060

Virtual IP address configuration

Virtual IP Addressing (VIPA) frees other hosts from depending on a particular physical network interface for communication with a z/VM TCP/IP stack.

The interface-layer fault-tolerance feature can be used with VIPA addresses, where applications can target the VIPA address, and any failure of the real LAN hardware is handled by the interface-takeover function. This differs from traditional VIPA usage, where dynamic routing protocols are required to route around real hardware failures. Example 3-6 illustrates the configuration of VIPA in the z/VM TCP/IP configuration file.

Example 3-6 z/VM Virtual IP Address configuration in TCPIP PROFILE

```
[...]
DEVICE DEV@2040 OSD 2040 NONROUTER
LINK OSA2040 QDIOETHERNET DEV@2040 NOPATHMTU MTU 4096 IP
DEVICE DEV@2060 OSD 2060 NONROUTER
LINK OSA2060 QDIOETHERNET DEV@2060 NOPATHMTU MTU 4096 IP
DEVICE DEVVIPA VIRTUAL 0
LINK OSAVIPA VIRTUAL 0 DEVVIPA
; (End DEVICE and LINK statements)
; -----
; -----
HOME
9.12.7.98 255.255.240.0 OSAVIPA
9.12.5.59 255.255.240.0 OSA2040
9.12.7.112 255.255.240.0 OSA2060
; (End HOME Address information)
[...]
```

Dedicated OSA connections to Linux machines

For performance, compliance, or security reasons, some implementations require dedicated, distinct, network connections to the outside world. In this case, OSA devices will be dedicated to a Linux machine, either running in an LPAR or as a guest of z/VM.

Ethernet bonding driver

To ensure network service continuity in case of hardware or link failures and remove a potential single point of failure, at least two sets of OSA devices have to be allocated to the Linux machine, using at least two different OSA cards. These sets of devices can then be logically aggregated together using a specific Linux driver, the Ethernet bonding driver, to create a single network interface in Linux, as shown in Figure 3-13.

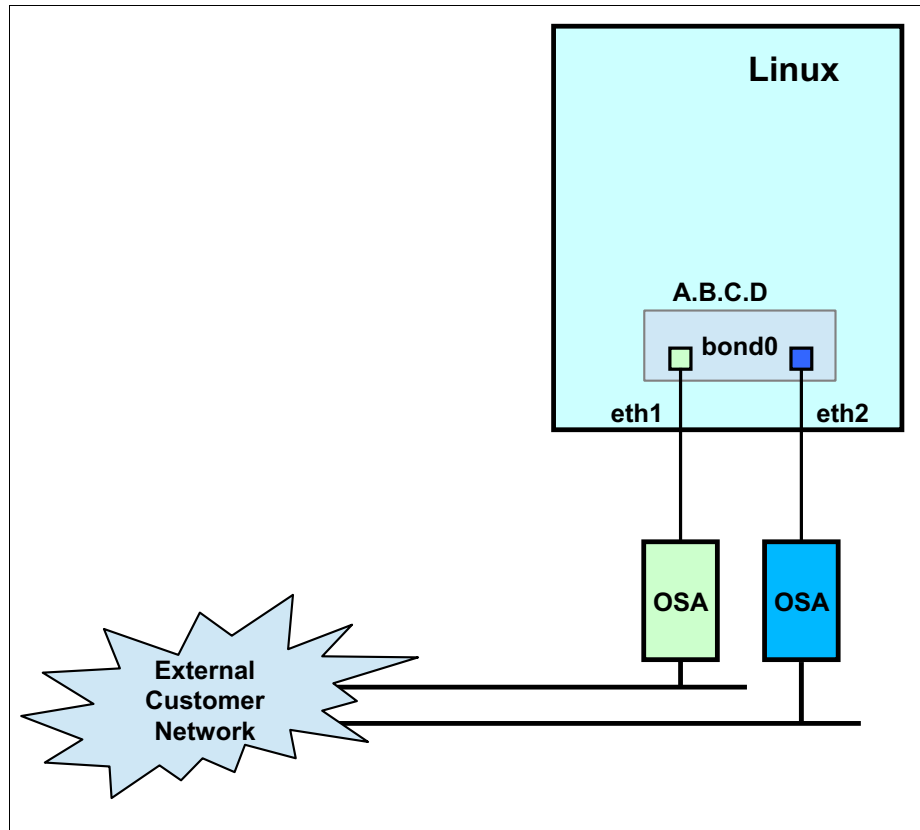


Figure 3-13 Linux bonding driver

Note: Some documentation also refers to this driver as the Channel Bonding driver.

The bonding driver in Linux can be used for either high-availability, load-balancing, or both, depending on its configuration. The default policy, called `balance-rr`, provides load-balancing and fault tolerance. The different configuration modes are as follows:

- ▶ `balance-rr` (or 0): default policy.
Packets are sent sequentially through each interface enslaved in the bonding interface, in order. This policy provides load balancing and fault tolerance.
- ▶ `active-backup` (or 1).
Only one enslaved interface is active, the others acting as backup interfaces. Another slave interface becomes active when the active one fails. This policy provides fault tolerance.
- ▶ `balance-xor` (or 2).
Slave to use for transmission is chosen according to the transmit hash policy. This policy provides load balancing and fault tolerance.

- ▶ broadcast (or 3).
Transmit everything on all slave interfaces. This policy provides fault tolerance.
- ▶ 802.3ad (or 4).
IEEE 802.3ad Dynamic Link Aggregation
- ▶ balance-tlb (or 5).
Adaptative Transmit Load Balancing. Outgoing traffic is distributed according to the load on each of the slave in the bond interface. This policy provides load balancing.
- ▶ balance-alb (or 6).
Adaptative Load Balancing. This policy includes the balance-tlb policy together with a Receive Load Balancing Policy. This policy provides load balancing.

Note: For the bonding driver to work, the underlying network interfaces have to operate at the Link Layer level, also known as Layer 2, or Ethernet layer. The bonding driver policies rely on MAC addresses to operate their choices. Therefore, a proper MAC addresses management policy has to be in place to avoid MAC addresses duplication over the network when working with dedicated OSA devices.

Another mandatory parameter is `miimon`, used to specify how often the link state for each slave interface should be checked. The default value is 0, meaning the link state is never checked. The documentation mentions 100 as being a good starting point, but this value may require further adjustment based on the network environment.

For a complete reference and documentation of all available options, refer to the Linux Ethernet Bonding Driver HOWTO as found in the kernel-source package, or online at the following URL:

<https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/networking/bonding.txt>

Configuring the bonding driver in SUSE Linux Enterprise Server 11 SP 3

YaST2 (Yet Another Setup Tool 2), a SUSE dedicated configuration tool, has a specific module for the configuration of bonding interfaces. The SUSE Linux Enterprise High Availability Extension documentation contains all the documentation and can be found here:

https://www.suse.com/documentation/sle_ha/book_sleha/data/cha_ha_netbonding.html

It takes three steps to configure a bonding interface in SUSE SLES11 SP3, after all hardware definitions and allocations are done:

1. Configure the OSA card to operate in Layer 2 mode. This is done by setting the Layer 2 parameter to 1 in the udev rules definition file, as shown in Example 3-7.

Example 3-7 Hardware configuration file for enslaved network interface in SUSE SLES11 SP3

```
lnxdisk:~ # cat /etc/udev/rules.d/51-qeth-0.0.2060.rules
# Configure qeth device at 0.0.2060/0.0.2061/0.0.2062
ACTION=="add", SUBSYSTEM=="drivers", KERNEL=="qeth", IMPORT{program}="collect
0.0.2060 %k 0.0.2060 0.0.2061 0.0.2062 qeth"
ACTION=="add", SUBSYSTEM=="ccw", KERNEL=="0.0.2060", IMPORT{program}="collect
0.0.2060 %k 0.0.2060 0.0.2061 0.0.2062 qeth"
ACTION=="add", SUBSYSTEM=="ccw", KERNEL=="0.0.2061", IMPORT{program}="collect
0.0.2060 %k 0.0.2060 0.0.2061 0.0.2062 qeth"
ACTION=="add", SUBSYSTEM=="ccw", KERNEL=="0.0.2062", IMPORT{program}="collect
0.0.2060 %k 0.0.2060 0.0.2061 0.0.2062 qeth"
```

```

ACTION=="remove", SUBSYSTEM=="drivers", KERNEL=="qeth",
IMPORT{program}="collect --remove 0.0.2060 %k 0.0.2060 0.0.2061 0.0.2062 qeth"
ACTION=="remove", SUBSYSTEM=="ccw", KERNEL=="0.0.2060",
IMPORT{program}="collect --remove 0.0.2060 %k 0.0.2060 0.0.2061 0.0.2062 qeth"
ACTION=="remove", SUBSYSTEM=="ccw", KERNEL=="0.0.2061",
IMPORT{program}="collect --remove 0.0.2060 %k 0.0.2060 0.0.2061 0.0.2062 qeth"
ACTION=="remove", SUBSYSTEM=="ccw", KERNEL=="0.0.2062",
IMPORT{program}="collect --remove 0.0.2060 %k 0.0.2060 0.0.2061 0.0.2062 qeth"
TEST=="[ccwgroup/0.0.2060]", GOTO="qeth-0.0.2060-end"
ACTION=="add", SUBSYSTEM=="ccw", ENV{COLLECT_0.0.2060}=="0",
ATTR{[drivers/ccwgroup:qeth]group}="0.0.2060,0.0.2061,0.0.2062"
ACTION=="add", SUBSYSTEM=="drivers", KERNEL=="qeth",
ENV{COLLECT_0.0.2060}=="0",
ATTR{[drivers/ccwgroup:qeth]group}="0.0.2060,0.0.2061,0.0.2062"
LABEL="qeth-0.0.2060-end"
ACTION=="add", SUBSYSTEM=="ccwgroup", KERNEL=="0.0.2060", ATTR{portno}="0"
ACTION=="add", SUBSYSTEM=="ccwgroup", KERNEL=="0.0.2060", ATTR{layer2}="1"
ACTION=="add", SUBSYSTEM=="ccwgroup", KERNEL=="0.0.2060", ATTR{online}="1"

```

Note: Failure to configure the OSA card to operate in layer 2 mode results in a non-functional bonding configuration.

2. Configure the slave interfaces, as shown in Example 3-8.

Example 3-8 Slave interface eth1 configuration file (ifcfg-eth1)

```

BOOTPROTO='none'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR=''
MTU=''
NAME='OSA Express Network card (0.0.2060)'
NETMASK=''
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'
LLADDR=e4:1f:00:00:00:01

```

3. Configure the bonding interface itself. The configuration file /etc/sysconfig/network/ifcfg-bond0 is shown in Example 3-9.

Example 3-9 SUSE SLES11 SP3 ifcfg-bond0 configuration example

```

BONDING_MASTER='yes'
BONDING_MODULE_OPTS='mode=active-backup miimon=100'
BONDING_SLAVE0='eth1'
BONDING_SLAVE1='eth2'
BOOTPROTO='static'
BROADCAST=''
ETHTOOL_OPTIONS=''
MTU=''
NAME=''
NETWORK=''
REMOTE_IPADDR=''

```



```
STARTMODE='auto'  
USERCONTROL='no'
```

The interface is specified as a bonding master interface, including a number of slave interfaces, which are specified with the **BOND_SLAVE*** statements. The bonding interface is configured using the active-backup policy, and link state for each slave is inspected every 100 ms as per the **miimon** parameter.

Configuring the bonding driver in Red Hat Enterprise Linux 6.5

In RHEL 6.5, it only takes two steps after the hardware definitions and allocations are done.

1. Each of the slave network interfaces needs to be configured as shown in Example 3-10.

Example 3-10 Enslaved network interface configuration file in Red Hat Enterprise Linux

```
[root@lnxrh1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-eth1  
DEVICE="eth1"  
BOOTPROTO="none"  
MASTER="bond0"  
NETTYPE="qeth"  
NM_CONTROLLED="no"  
USERCTL="no"  
ONBOOT="yes"  
OPTIONS="layer2=1"  
SUBCHANNELS="0.0.2060,0.0.2061,0.0.2062"  
SLAVE="yes"  
MACADDR="e4:1f:00:00:00:01"
```

2. The bonding interface, bond0, is then configured using the file shown in Example 3-11 as an example.

Example 3-11 Bonding interface configuration file in Red Hat Enterprise Linux

```
[root@lnxrh1 ~]# cat /etc/sysconfig/network-scripts/ifcfg-bond0  
DEVICE=bond0  
IPADDR=9.12.7.91  
NETMASK=255.255.252.0  
ONBOOT=yes  
BOOTPROTO=static  
USERCTL=no  
NM_CONTROLLED=no  
BONDING_OPTS="mode=active-backup miimon=100"  
BROADCAST="9.12.7.255"  
DNS="9.12.6.7"  
GATEWAY="9.12.4.1"
```

Note: In Red Hat Enterprise Linux 7, the bonding driver is replaced by the team driver. A migration tool, **bond2team**, exists to assist customers in migrating their configurations from using the bonding driver to the team driver.

Example 3-12 on page 60 shows the bonding configuration results. The two enslaved interfaces, eth1 and eth2, are configured from a hardware perspective only. They do not carry any useful network configuration. Network configuration is done for the bond0 interface, which carries the IP address, netmask, and all other relevant network configuration statements.

Example 3-12 Completed Linux bonding configuration in SUSE Linux Enterprise Server

```
lnxdisk:~ # ifconfig -a
bond0      Link encap:Ethernet  HWaddr E4:1F:00:00:00:01
            inet addr:9.12.7.91  Bcast:9.12.7.255  Mask:255.255.252.0
            inet6 addr: fe80::e61f:ff:fe00:1/64 Scope:Link
            UP BROADCAST RUNNING MASTER MULTICAST  MTU:1500  Metric:1
            RX packets:51722 errors:0 dropped:1891 overruns:0 frame:0
            TX packets:24158 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:66961217 (63.8 Mb)  TX bytes:1748182 (1.6 Mb)

eth1       Link encap:Ethernet  HWaddr E4:1F:00:00:00:01
            UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
            RX packets:49556 errors:0 dropped:2 overruns:0 frame:0
            TX packets:24158 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:66841337 (63.7 Mb)  TX bytes:1748182 (1.6 Mb)

eth2       Link encap:Ethernet  HWaddr E4:1F:00:00:00:01
            UP BROADCAST RUNNING SLAVE MULTICAST  MTU:1500  Metric:1
            RX packets:2166 errors:0 dropped:1871 overruns:0 frame:0
            TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:119880 (117.0 Kb)  TX bytes:0 (0.0 b)

lo         Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:16436  Metric:1
            RX packets:10 errors:0 dropped:0 overruns:0 frame:0
            TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:772 (772.0 b)  TX bytes:772 (772.0 b)
```

Note: Using the active-backup policy, the bonding interface will set all slave interfaces to use the same MAC address at enslavement time. The default is to use the MAC address of the first interface being enslaved.

Depending on the customer's network topology and requirements, this behavior can be changed using the `fail_over_mac` parameter.

Complete documentation for the bonding driver parameters can be found at the following website:

<https://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/Documentation/networking/bonding.txt>

The configuration can be explored through the `/sys` pseudo filesystem. Example 3-13 shows the layout of the relevant node in this filesystem (❶). The `slaves` node (❷) contains the list of enslaved interfaces, here `eth1` and `eth2`. The `active_slave` node (❸) contains the name of the interface which is currently the active slave, that is which interface is currently in use in the bond.

Example 3-13 Exploring the bonding configuration

```
lnxdisk:/sys/class/net/bond0/bonding # ls ❶
active_slave  ad_num_ports  ad_select      arp_ip_target  fail_over_mac
miimon        num_unsol_na   queue_id      updelay
ad_actor_key  ad_partner_key all_slaves_active arp_validate   lacp_rate
mode          primary        resend_igmp   use_carrier
ad_aggregator ad_partner_mac arp_interval   downdelay     mii_status
num_grat_arp  primary_reselect slaves         xmit_hash_policy
lnxdisk:/sys/class/net/bond0/bonding # cat slaves ❷
eth1 eth2
lnxdisk:/sys/class/net/bond0/bonding # cat active_slave ❸
eth1
lnxdisk:/sys/class/net/bond0/bonding #
```

Observing the amount of data transferred in the output of the **ifconfig** command (as shown in Example 3-12 on page 60) confirms that only eth1 is used, unless this link fails.

To ensure that the configuration worked as expected, we simulated a link failure on the active slave by issuing the **DETACH 2060-2062 LNXDISK** command during a large network transfer. The network transfer never got interrupted; the second slave interface transparently took over the work of the failed one.

3.6.3 z/VM Virtual Switch

The z/VM Virtual Switch (VSWITCH) is the most prevalent networking option used to connect Linux on System z guests to the outside world. It is a special type of guest LAN that can provide external LAN connectivity through Hipersockets or an OSA-Express device without the need for a routing virtual machine.

Default z/VM virtual switch configuration

All that the **DEFINE VSWITCH** command needs as parameters are the name of the VSWITCH to be created, as well as a set of real devices (RDEV) to be used for connecting to the outside network. Other parameters will take their default values when not specified.

However, it is possible to specify more than one set of real devices (up to three in the current VSWITCH implementation) in the **DEFINE VSWITCH** command, as shown in Example 3-14. One of these set of devices will act as the primary connection to the outside world, the other set or sets being backup. Transparent failover mechanisms are integrated into the VSWITCH to ensure seamless transition to the backup devices in case of a failure on the primary.

Example 3-14 Minimal command to create a VSWITCH

```
DEFINE VSWITCH VSWITCH1 RDEV 2043 2063
```

Note: The **DEFINE VSWITCH** command defaults to using the OSA-Express card port 0. However, the 4-port versions of OSA cards feature 2 OSA ports per CHPID. To use the second port of a given CHPID, the .P1 has to be specified after the real device address, as in 2043.P1, for instance.

The result of this command is a highly available VSWITCH, as shown in Example 3-15.

Example 3-15 Virtual switch VSWITCH1 definition

```
q vswitch vswitch1
VSWITCH SYSTEM VSWITCH1 Type: QDIO      Connected: 2      Maxconn: INFINITE
  PERSISTENT  RESTRICTED  NONROUTER      Accounting: OFF
  USERBASED
  VLAN Unaware
  MAC address: 02-00-0C-00-00-01      MAC Protection: OFF
  ITimeout: 5      QueueStorage: 8
  Isolation Status: OFF      VEPA Status: OFF
  Uplink Port:
  State: Ready
  PMTUD setting: EXTERNAL  PMTUD value: 8992
  RDEV: 2043.P00 VDEV: 060F Controller: DTCVSW2 ACTIVE
    EQID: OSA1SET1
  RDEV: 2063.P00 VDEV: 060F Controller: DTCVSW1 BACKUP
    EQID: OSA1SET1
```

VSWITCH1 has been defined with 2 connections to the outside using 2 sets of real OSA devices:

- ▶ Devices 2043-2045, assigned as ACTIVE
- ▶ Devices 2063-2065, assigned as BACKUP

Note: It is necessary to allocate OSA devices from two different OSA cards to benefit from the VSWITCH built in high-availability features.

Upon creation, the VSWITCH is in a **READY** state, serving connections from the connected guests to the outside world. Devices 2043-2045 are assigned to VSWITCH controller DTCVSW2, and devices 2063-2065 are assigned to VSWITCH controller DTCVSW1.

Note: By default, z/VM ships with two virtual switch controllers, DTCVSW1 and DTCVSW2. Those virtual switch controllers are dedicated TCP/IP stacks for the management of the z/VM virtual switches. If necessary, additional virtual switches controllers can be added to the configuration, following instructions found in Usage Notes 3 for the **DEFINE VSWITCH** statement in the *z/VM: CP Commands and Utilities* book.

This setup isolates the VSWITCH from either VSWITCH controller failures or real OSA failures, as demonstrated in Example 3-16 and Example 3-17.

Example 3-16 Virtual switch VSWITCH1 behavior in case of VSWITCH controller failure

```
force dtcvsw2
USER DSC  LOGOFF AS  DTCVSW2  USERS = 33      FORCED BY MAINT
Ready; T=0.01/0.01 15:57:53
HCPSWU2843E The path was severed for TCP/IP Controller DTCVSW2.
HCPSWU2843E It was managing device 2063.P00 for VSWITCH SYSTEM VSWITCH1.
HCPSWU2845I Backup device 2063.P00 specified for VSWITCH VSWITCH1 is not initial
ized.
RDR FILE 0183 SENT FROM DTCVSW2  CON WAS 0005 RECS 0061 CPY 001 T NOHOLD NOKEEP
HCPSWU2830I DTCVSW1 is VSWITCH controller for backup device 2063.P00.

q vswitch vswitch1
```

```

VSWITCH SYSTEM VSWITCH1 Type: QDIO    Connected: 2    Maxconn: INFINITE
PERSISTENT RESTRICTED    NONROUTER    Accounting: OFF
USERBASED
VLAN Unaware
MAC address: 02-00-0C-00-00-01    MAC Protection: OFF
IPTimeout: 5    QueueStorage: 8
Isolation Status: OFF    VEPA Status: OFF
Uplink Port:
State: Ready
PMTUD setting: EXTERNAL    PMTUD value: 8992
RDEV: 2043.P00 VDEV: 0612 Controller: DTCVSW1    ACTIVE
EQID: OSA1SET1
RDEV: 2063.P00 VDEV: 060F Controller: DTCVSW1    BACKUP
EQID: OSA1SET1
Ready; T=0.01/0.01 15:51:12

```

In case of a link failure, the backup link takes over, as illustrated in Example 3-17 when we detached the OSA devices from the active VSWITCH controller DTCVSW2.

Example 3-17 Virtual switch VSWITCH1 behavior in case of link failure

```

q vswitch vswitch1
VSWITCH SYSTEM VSWITCH1 Type: QDIO    Connected: 0    Maxconn: INFINITE
PERSISTENT RESTRICTED    NONROUTER    Accounting: OFF
USERBASED
VLAN Unaware
MAC address: 02-00-0C-00-00-23    MAC Protection: OFF
IPTimeout: 5    QueueStorage: 8
Isolation Status: OFF    VEPA Status: OFF
Uplink Port:
State: Ready
PMTUD setting: EXTERNAL    PMTUD value: 8992
RDEV: 2043.P00 VDEV: 0603 Controller: DTCVSW2    ACTIVE
EQID: OSA1SET1
RDEV: 2063.P00 VDEV: 061E Controller: DTCVSW1    BACKUP
EQID: OSA1SET1
Ready; T=0.01/0.01 16:02:09
det 2043-2045 dtcvs2
2043-2045 DETACHED DTCVSW2
HCPSWU2832E Connection 2043.P00 for VSWITCH SYSTEM VSWITCH1 is not active.
HCPSWU2832E Device is detached from DTCVSW2.
HCPSWU2830I VSWITCH SYSTEM VSWITCH1 status is in error recovery.
HCPSWU2830I DTCVSW1 is new VSWITCH controller for device 2063.P00.
Ready; T=0.01/0.01 16:03:00
HCPSWU2830I VSWITCH SYSTEM VSWITCH1 status is ready.
HCPSWU2830I DTCVSW1 is VSWITCH controller for device 2063.P00.

```

```

q vswitch vswitch1
VSWITCH SYSTEM VSWITCH1 Type: QDIO    Connected: 0    Maxconn: INFINITE
PERSISTENT RESTRICTED    NONROUTER    Accounting: OFF
USERBASED
VLAN Unaware
MAC address: 02-00-0C-00-00-23    MAC Protection: OFF
IPTimeout: 5    QueueStorage: 8
Isolation Status: OFF    VEPA Status: OFF
Uplink Port:

```

```

State: Ready
PMTUD setting: EXTERNAL   PMTUD value: 8992
RDEV: 2043.P00 VDEV: NONE Controller: NONE
EQID: OSA1SET1
RDEV: 2063.P00 VDEV: 061E Controller: DTCVSW1 ACTIVE
EQID: OSA1SET1
Ready; T=0.01/0.01 16:03:34

```

To further improve the availability and performance of the virtual switch, the link aggregation feature of the z/VM virtual switch can be implemented instead.

z/VM virtual switch link aggregation

The virtual switch can be configured to operate with aggregated links in concert with a switch that also supports the IEEE802.3ad Link Aggregation specification, as shown in Figure 3-14. Aggregating links with a partner switch box allows multiple OSA-Express5S, OSA-Express4S, OSA-Express3, or OSA-Express2 adapters to be deployed in transporting data between the switches. This configuration provides the benefits of increased throughput and nearly seamless recovery of a failed link within the aggregated group.

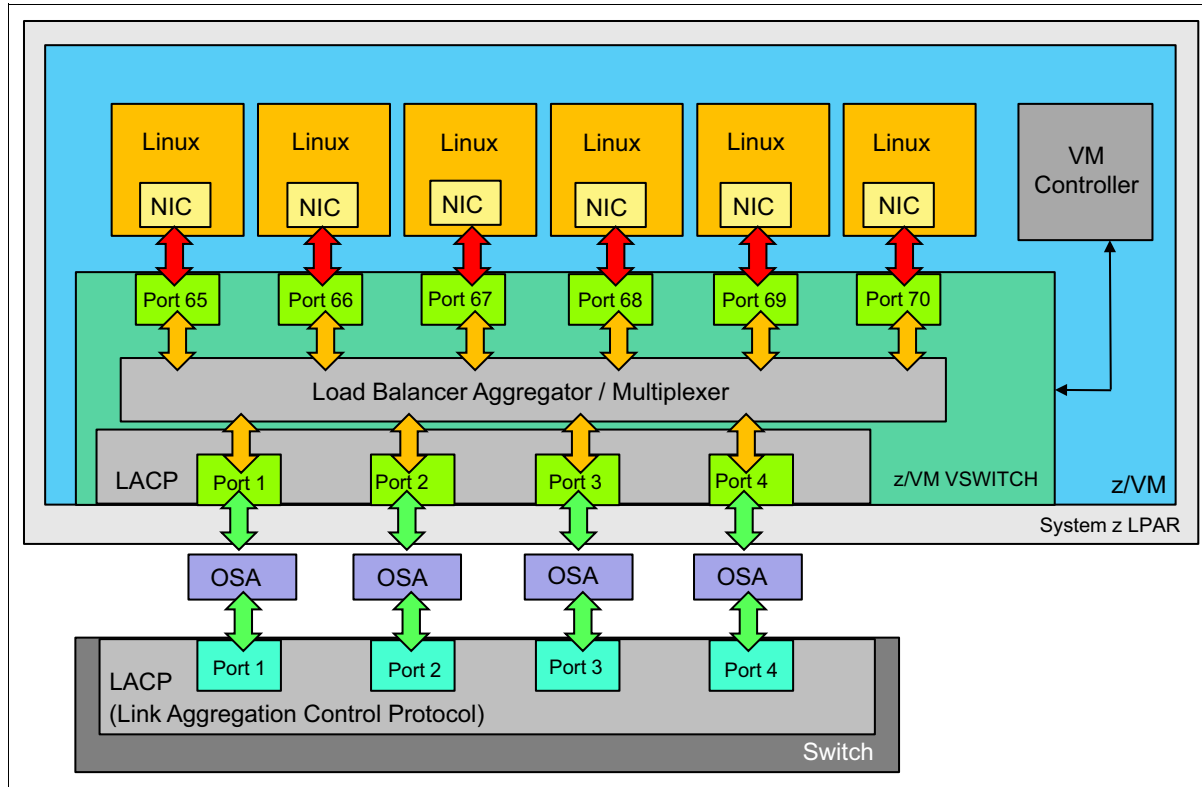


Figure 3-14 z/VM virtual switch Link Aggregation

Link aggregation for z/VM in Layer 2 mode functions as follows:

- ▶ Aggregated links are viewed as one logical trunk and contain all of the virtual LANs (VLANs) required by the LAN segment.
- ▶ Link aggregation is between a VSWITCH and the physical network switch.
- ▶ Load-balanced communications are across multiple links in a trunk to prevent a single link from being overrun.

- ▶ Up to eight OSA-Express5S, OSA-Express4S, OSA-Express3, or OSA-Express2 are supported in one aggregated link.
- ▶ Ability to dynamically add/remove OSA ports for on demand bandwidth.

z/VM virtual-switch-controlled (VSWITCH-controlled) link aggregation (IEEE3 802.3ad) support requirements include these:

- ▶ Layer 2 virtual switches, that is defined with the ETHERNET (default) option, and using guests that support Layer 2.
- ▶ OSA devices must support IEEE802.3ad Link Aggregation. OSA-Express 2 and higher have this support.
- ▶ All NIC cards must have the same data rate specification support (10MBps,100MBps,1GBps, 10GBps).
- ▶ OSA cards are not shared with another z/VM or LPAR. The OSA cards used for port aggregation have to be exclusively dedicated for this use.
- ▶ Up to eight OSA cards can be aggregated per virtual switch.
- ▶ Each OSA-Express card is directly and solely connected to the same switch.
- ▶ Target OSA-Express devices must be in full-duplex mode.
- ▶ When VLANs are deployed over the aggregated link, all member OSA links must be trunk links to provide the virtual LAN connectivity in which to flow tagged traffic. The aggregated link should be viewed as one logical trunk link containing all the VLANs required by the LAN segment.
- ▶ When the port group is configured as LACP inactive, all ports of the group must support the network disablement function of the OSA-express feature. Cards that do not support the network disablement function may be used only in a LACP active group. The network disablement function allows the virtual switch to inform the partner switch that a port is no longer functional and allows the partner switch to react and perform port recovery.

z/VM virtual switch port aggregation configuration

Assuming all network configuration is in place, configuring link aggregation requires the following steps:

1. Create a port group using the **SET PORT** command, as shown in Example 3-18.

Example 3-18 Create a port group for z/VM to use in a virtual switch

```
SET PORT GROUP ETHGRP 2060 204F
```

2. Associate the newly created port group to the VSWITCH, as described in Example 3-19.

Example 3-19 Associate the port group to an already existing virtual switch VSWITCH1

```
SET VSWITCH VSWITCH1 DISCONNECT
SET VSWITCH VSWITCH1 GROUP ETHGRP
SET VSWITCH VSWITCH1 CONNECT
```

For more details about the configuration and use of the z/VM Virtual Switch Link Aggregation feature, see the following documents:

- ▶ z/VM Virtual Switch Link Aggregation website at this website:
<http://www.vm.ibm.com/virtualnetwork/linkag.html>
- ▶ The “Link Aggregation” chapter of the z/VM: *Connectivity* book, SC24-6174-04.

3.6.4 HiperSockets

HiperSockets is an extension of the Queued Direct IO (QDIO) hardware facility that provides a microcode-only vehicle for IP inter-program communications. Communications between programs is accomplished with TCP/IP socket connections. Using HiperSockets, a program has the ability not only to directly communicate with a program running in the same LPAR, but also to communicate across any logical partition within the same machine.

As a microcode-only feature, a HiperSockets network is considered to be highly available by design.

Bridged HiperSockets channel

A HiperSockets channel by itself is only capable of providing intra-CEC (Central Electronic Complex) communications. The HiperSockets Bridge Port allows a virtual switch to connect z/VM guests using real HiperSockets devices, the ability to communicate with hosts residing outside of the CEC. A single IP address and virtual machine network connection can be used to communicate over the internal and external segments of the lan. The fact that a given destination address may reside on the local HiperSockets channel or outside the CEC is totally transparent to the bridge capable port.

HiperSockets bridge high availability

To ensure continuous network availability for a bridged HiperSockets channel, it is advised that every z/VM LPAR that has bridge capable guests connected to the HiperSockets channel, also have a virtual switch connected to the same layer 2 network with a secondary Bridge Port in standby status, as shown in Figure 3-15.

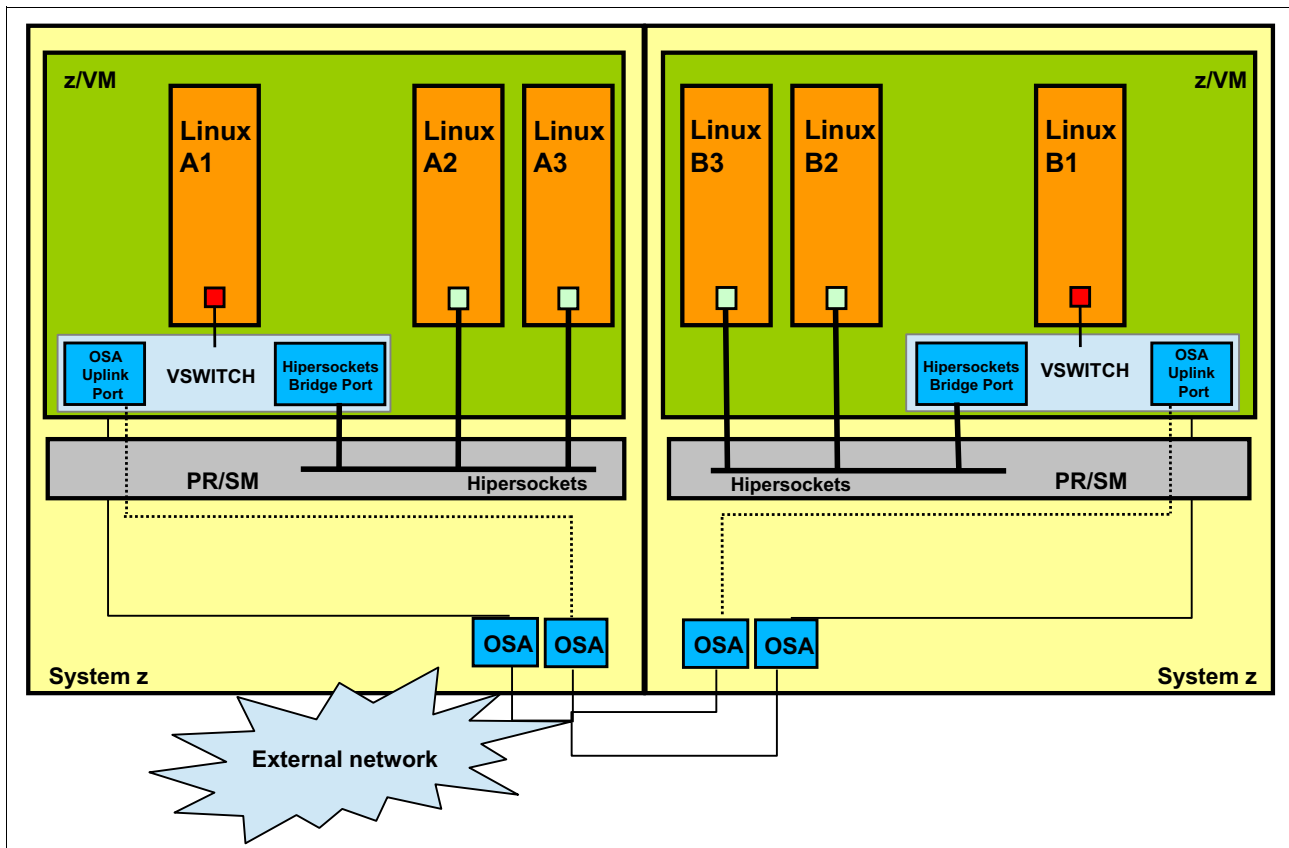


Figure 3-15 z/VM HiperSockets Bridge

Two layers of failover operations that are in place for a bridged HiperSockets channel ensure sustained connectivity for the bridged channel:

- **HiperSockets Directed Failover Operation**

When the HiperSockets firmware detects that the currently active Bridge Port is non-functional, it will immediately initiate an adapter initiated state change to place the subject Bridge Port into inactive status and select a Bridge Port currently in standby status and make it active. After being notified of its change to active status, the virtual switch Bridge Port assumes the bridging function for the HiperSockets channel. This unanticipated recovery action or system administrator initiated configuration change commences transparently to all HiperSockets bridge capable guest ports.

Note that if the failing Bridge Port is assigned the role of primary, then upon its subsequent recovery, it will be returned from standby status to active status. If the failing Bridge Port was assigned a secondary role, then the HiperSockets firmware will maintain the current active configuration. If the secondary Bridge Port that went down recovers, it will be placed in standby status.

- **Virtual Switch Directed Failover Operation (as described in “Default z/VM virtual switch configuration” on page 61)**

The virtual switch has indigenous recovery operations for both OSA-Express failures and controller failures. Uplink port (RDEV) redundancy is dependent on additional resources (RDEVs) being available (configured). An additional controller is already provided with z/VM to be deployed by the virtual switch as needed. Additional controllers may be created by the z/VM System Administrator as needed.



Architectural scenarios

High availability solutions always involve redundancy, and the basic rule of high availability is to identify and remove single points of failure (SPOFs) in the architecture. A user environment can have multiple layers, such as, cloud, application, firewall, security, facilities, storage, and database layers. All of these layers can be in one or multiple data centers, and they can be in one or multiple servers that are running under multiple operating systems.

In Figure 4-1 on page 70, we see various components of a database node as part of a Linux on System z environment. Each one of the components can be identified as an SPOF. It is important that all SPOFs be eliminated, so that the surviving components will continue to run in the event that one or more components fail.

Usually a single data center architecture is supported, but it is not a true highly available architecture. If the entire data center fails, there will be no automated failover. This architecture is only valid for protecting data through data replication, and for protecting against multiple system node failures.

In this chapter, we show a two-site architecture that would eliminate each of the SPOFs using various technologies already discussed in the previous chapters.

This chapter covers the following topics:

- ▶ Database high availability: Two sites
- ▶ z/OS database scenario
- ▶ z/OS and Linux scenarios

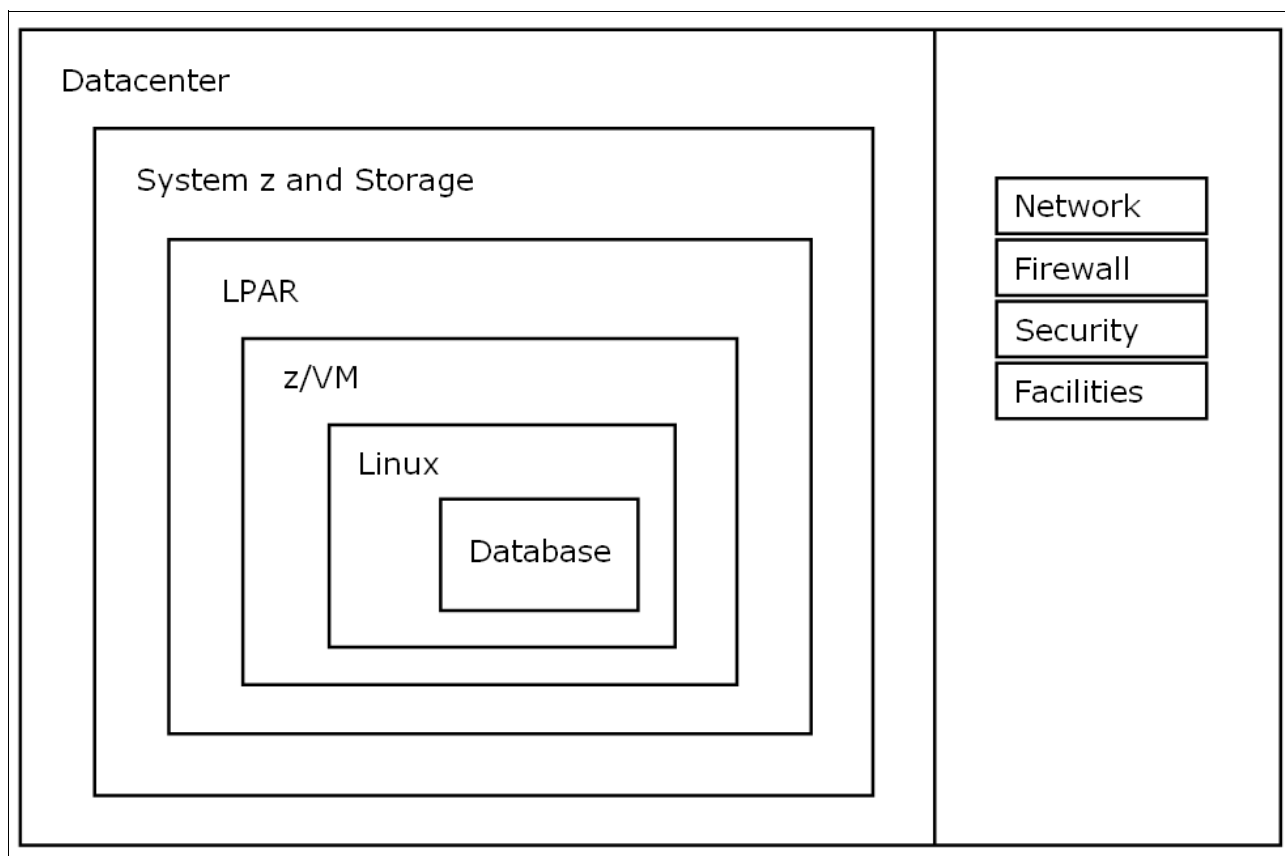


Figure 4-1 Components that could be a source of a single point of failure

4.1 Database high availability: Two sites

In this section, we discuss a two-site database solution on System z (Figure 4-2) along with other components. The System z solution provides a combination of availability, reliability, security and resilience, together with a unique ability to reallocate processing power to match changing business priorities during peak demand times.

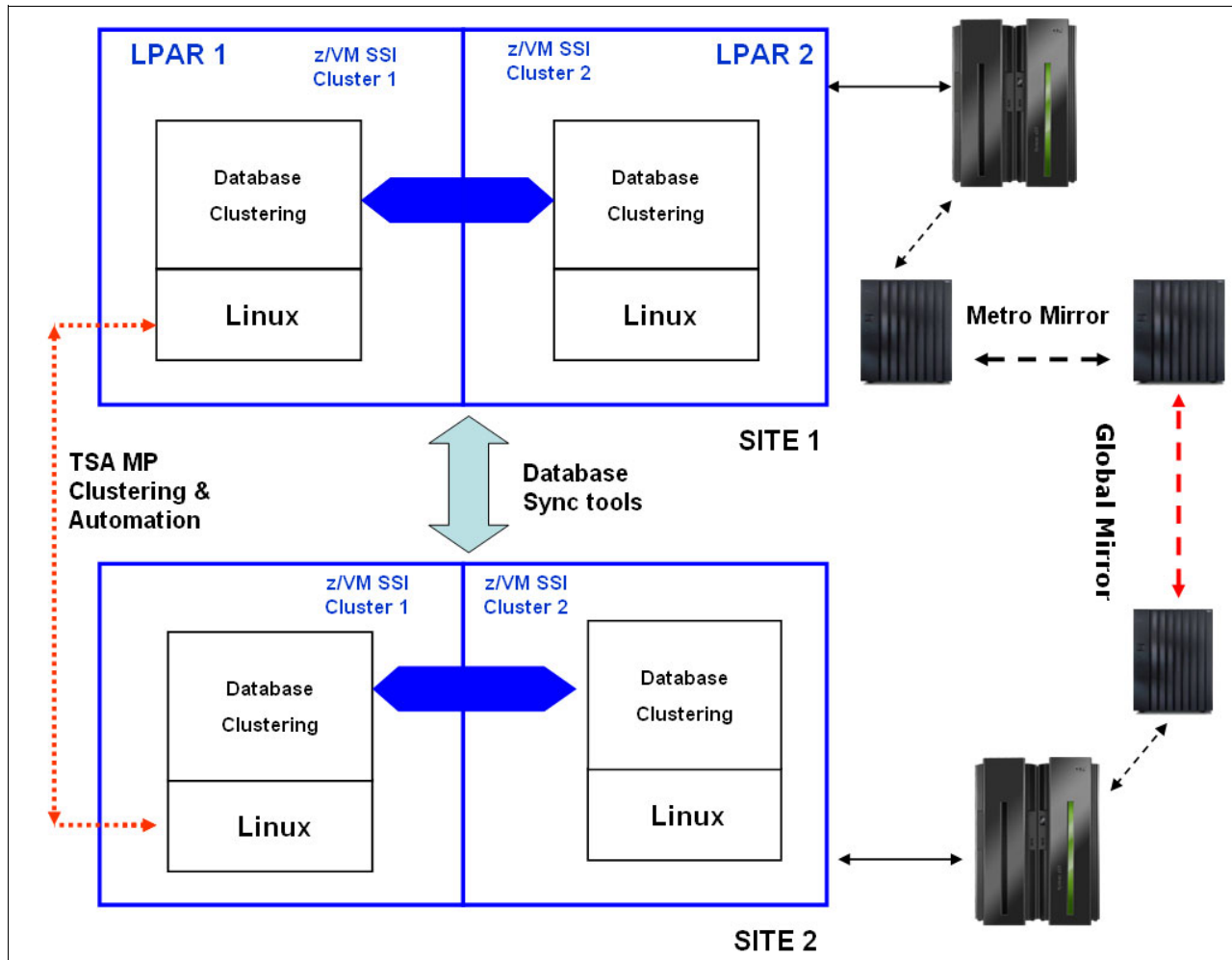


Figure 4-2 Two-site database architectural scenario

In this section, we also discuss each single point of failure (SPOF) and how it would be eliminated by each example architectural scenario.

Datacenter layer

In Figure 4-2, the data center is the top layer where all the components that are needed for databases on Linux on System z are running. A datacenter can be classified as Multiple Points of Failure (MPoFs), because any failure would cause many components to fail, such as the failure of a data center, of an entire site, or of a small area. A data center, in this context encompasses all of the servers, storage, network, facility, human resources, software, and other components that are required to run the databases.

Power and cooling

In the data center layer, having multi-site datacenters and levels of power and cooling equipment redundancies can mitigate the impact of a facilities based hardware failure. Power for each data center is supplied from different power circuits. Some may even want to lease redundant power from different substations.

Network and storage links

It is also expected that the networks and storage links (including switches) between the data centers are redundant and routed in such a way that the loss of any one data center does not cause the network between surviving data centers to fail.

Apart from this, Heartbeat networks (non-IP) between the data center must be redundant and routed along physically different paths so that the loss of any one site or data center does not cause the network between surviving data centers to fail. All the nodes using clustering software must be configured to use the Heartbeat network.

However, the architecture has to mitigate the overall availability of the IT resources, by means of redundant servers, storage and networks, as well as the software to monitor, manage and re-allocate and re-direct applications and processes to other resources in the event of an IT system failure.

System z hardware and storage layer

Ensure that the System z and storage hardware is redundant. This eliminates single points of failure (SPOFs) on the hardware layer of the architecture. Apart from physical hardware, we have also eliminated the SPOFs associated with logical partitions (LPARs). This is done by creating a second LPAR that will run IBM z/VM as a hypervisor. The cost for doing this is still low, because both LPARs can share the same processors, and the real memory can be split between the two LPARs.

In this scenario, the primary database storage is replicated to the second site using synchronous Metro Mirror Geographically Dispersed Parallel Sysplex Peer to Peer Remote Copy (GDPS PPRC), maintaining data consistency between the primary and secondary devices on the two sites.

In case of a site failure, the application needs to be redirected to the secondary database server and also the consistency of the database needs to be validated before activating the application connectivity. This method requires a manual intervention for activation and verification of the database server.

The Metro-Mirror (GDPS PPRC) standby data would be consistent depending on whether or not the application or database vendor supports a PPRC environment. In this case there may be inconsistencies in the database. This is due to the fact that the database server is not doing the replication and it is a pure storage based replication. The recovery at the secondary site cannot be automated for that scenario and is done manually. Hence the RTO would be the time required to manually restart the machine from the DR site and bring the DB and applications up.

Operating system layer

To eliminate SPOFs, you must also look at the operating system layer. In this section, we describe how we eliminate SPOFs in the z/VM cluster as well as on Linux servers.

z/VM cluster

We have eliminated the SPOFs on the z/VM Hypervisor by clustering the second z/VM LPAR. Even though currently z/VM Single System Image (SSI) clustering does not provide high

availability features, this feature can be used effectively during planned outages. In case of planned downtime of the logical partition, the Linux servers can be moved to other z/VM LPARs using live guest relocation.

Linux on System z and clustering

A failure in any Linux server still allows the application to run with the full resources available to it before on the remaining virtual servers. This feature is unique to Linux on System z. Because all of the CPUs are shared among the production and standby LPAR, a failure of one Linux server or application frees up its CPU for use by other Linux servers in the standby LPAR. For example, if two application servers are both 80% CPU utilized (using 80% of one CPU), if one of them fails, the other can increase its CPU utilization to 80% of the two CPUs. This allows the remaining server to process the full workload immediately, without having to reconfigure the server.

4.1.1 High availability clustering and automation tools

An active/passive configuration consists of a server that owns the cluster resources and other servers that are capable of accessing the resources that are on standby until the cluster resource owner is no longer available. The advantages of the active/passive configuration are that there is no service degradation and services are only restarted when the active server no longer responds.

High availability architecture, as shown in Figure 4-3, can be achieved using the IBM Tivoli System Automation for Multiplatforms (SA) and the open source alternative of Linux-HA. We address the methodology for using both the tools in this section.

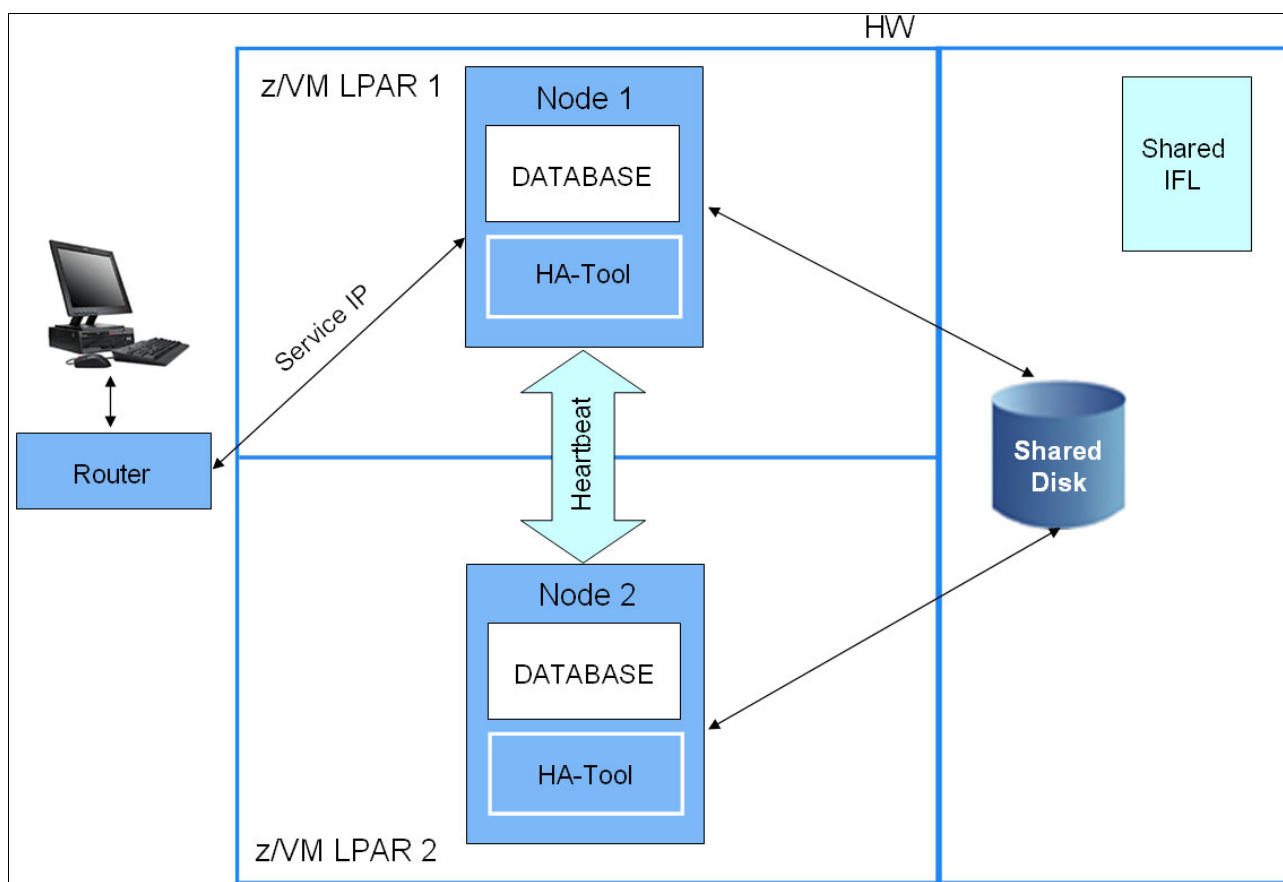


Figure 4-3 Active passive setup

Using Tivoli System Automation

IBM Tivoli System Automation for Multiplatforms (SA MP) runs on each node in the cluster. It monitors cluster nodes and exchanges information through Reliable Scalable Cluster Technology (RSCT) services.

Using Figure 4-3 on page 73 as an example, SA MP creates a service IP address as an alias on an appropriate network adapter on Node 1 (where perhaps an Oracle server will be started). One and only one instance of the Oracle server is defined to SA MP to be able to run in either of the two nodes, with a “depends on” relationship to a single IP address (the service IP). SA MP starts the database server on Node 1, then at user-defined intervals, invokes a script to confirm that it is still up and serving pages. It also monitors the Linux node itself to ensure that it remains active. In case of an Oracle server node crash, RSCT determines that Node 1 is no longer responding. Sa MP then moves the service IP over to Node 2, and starts the Oracle server there.

Using Linux-HA

The Linux High Availability (Linux-HA) project provides high availability solutions for Linux through an open source development community. The majority of Linux-HA software is licensed under the Free Software Foundation’s GNU General Public License (GPL) and the Free Software Foundation’s GNU Lesser General Public License (LGPL). For more licensing information, see this website:

<http://www.linux-ha.org/LegalInfo>

Linux-HA is a standard, robust open-source package for creating Linux clusters. Linux-HA is packaged with other tools from different distributors to form a highly available solution.

The Linux-HA release 2 software package provides the following capabilities:

- ▶ Active/active and active/passive configurations
- ▶ Failover and failback on node, IP address, or resource failure
- ▶ Failover and failback on customized resource
- ▶ Support for the Open Cluster Framework (OCF) resource standard and Linux Standard Base (LSB) resource specification
- ▶ Both command line interface and graphical user interface for configuration and monitoring
- ▶ Support for up to a 16-node cluster
- ▶ Multi-state (master/slave) resource support
- ▶ Rich constraint support
- ▶ XML-based resource configuration
- ▶ Has no kernel or hardware dependencies
- ▶ Load balancing capabilities with Linux Virtual Server (LVS)

The core of Linux-HA release 2 is a component called Heartbeat. Heartbeat provides the clustering capability that ensures high availability of critical resources such as data, applications, and services. It does this by providing monitoring, failover, and failback capabilities to Heartbeat defined resources.

The failover solution allows clients to know about the presence of peer processes on other machines and to easily exchange messages with them. The Heartbeat tool needs to work in conjunction with other cluster managers (such as Pacemaker) to manage a cluster (initiate or kill process).

After it is configured, the Heartbeat tool running on the standby server can check for pings (heartbeats) coming from the primary server over the normal Ethernet network connection. But it is normally recommended to configure Heartbeat nodes with two network interfaces (service and standby) and in addition, a non-IP heartbeat connection. Also, z/VM and Linux support both IP and non-IP based protocols (for example, IPv4, IPv6). An additional non-IP connection is preferred because it does not rely on the IP stack, so a failure of the network adapter can be identified.

In case of a primary server failure, Heartbeat uses a method called IP Address Takeover (IPAT), to move the IP address from the primary computer to the backup computer. To accomplish IPAT, Heartbeat uses secondary IP addresses and gratuitous ARP broadcasts.

For more information about Heartbeat and Linux HA, see the IBM Redbooks publication, *Achieving High Availability on Linux for System z with Linux-HA Release 2*, SG24-7711.

4.1.2 Database layer

This architecture involves identically configured database environments on the primary and secondary sites. The primary site contains a production database using vendor specific cluster tools to protect them from host and instance failures.

Using database replication tools

In the two-site option, the database replication features (for example, Oracle Dataguard or IBM DB2® HADR) perform a synchronous database log replication between the two sites for the databases so that the primary site database is replicating with the DR site database. These database replication tools provide transaction integrity among the primary and secondary databases and if there is a disk failure, then the standby database will be current to the last committed transaction.

A partial site failure can be caused by a failure of hardware, network, or software (database system or operating system). Without database replication features, a partial site failure requires restarting the database management system (DBMS) server that contains the database. The length of time it takes to restart the database and the server where it resides is unpredictable. It can take several minutes before the database is brought back to a consistent state and made available. With DB2 HADR, the standby database can take over in seconds.

The application data typically would have very little changes and hence the procedure of End of Day (EoD) storage replication of application data between the primary and secondary databases can be performed.

In an Active/Active scenario using the inbuilt database, replication features might provide high availability to the databases, but it also would put a dent on performance, as the logs are continuously synched between the sites. In an Active/Passive scenario, even though the consistency of the database is preserved, there is a need for manual intervention to redirect the application access flow to the standby database.

Attention: Its important to note that depending on different database vendors, database clustering and replication tools have a predefined distance certification between the sites. As the distance between the primary and secondary databases increases, the performance of the database may decrease. Additionally, the vendor may not certify or provide support if database inconsistencies arise.

The type of high availability solution described in this section would be productive if the primary and secondary servers are located in the same site or near site locations. In the case of the primary and secondary servers being separated by a long distance (hundreds of miles), then the application performance may be affected.

4.2 z/OS database scenario

HA can be analyzed from site to application as discussed in 4.1, “Database high availability: Two sites” with a two-site database solution. Table 4-1 shows the difference between the z/OS platform and z/Linux platform. The main difference is the operating system and the features that come with it. This section illustrates the difference using a three-site solution with other products.

Table 4-1 Differences between z/OS solution and Linux on System z solution

	z/OS	z/Linux
Site	Same	Same
Hardware	Mainframe * z/OS uses CP * CFCC uses CP of ICF	Mainframe Linux uses CP or IFL
Operating system	z/OS CFCC (Micro code provide)	Linux on System z
Application database	DB2	DB2, Oracle
HA solution	Parallel Sysplex	Linux HA

4.2.1 Parallel Sysplex technology

Parallel Sysplex is a high-performance data sharing technology that relies on the use of a Coupling Facility (CF). The CF allows multiple processors to access the same data. The CF is implemented in a Logical Partition (LP) just like a z/OS image, but it runs its own highly specialized operating system, Coupling Facility Control Code (CFCC). It is connected to all the z/OS images in a Parallel Sysplex by high-speed fiber or copper links. The design of the CF ensures that requests are queued in a logical order that is not dependent on the speed of the connected central processor complex (CPC). This removes the unpredictable response time that is inherent to conventional devices such as DASD and channel to channel (CTC) communications.

XES is another component of z/OS. It provides the Parallel Sysplex services that applications and subsystems use to communicate with the Coupling Facility. These services support the sharing of cached data and common queues while maintaining data integrity and consistency. Any subsystem, such as DB2, that needs to perform activity against the CF structures uses XES services.

In the CF, data is organized into distinct objects called structures. CF structures are used by authorized programs to implement data sharing and high-speed serialization. There are a number of different types of CF structures that can be created, such as cache, list, and lock, each providing a specific function to the application. Some storage in the coupling facility can also be allocated as a dedicated dump space for capturing structured information for diagnostic purposes. The components are shown in Figure 4-4.

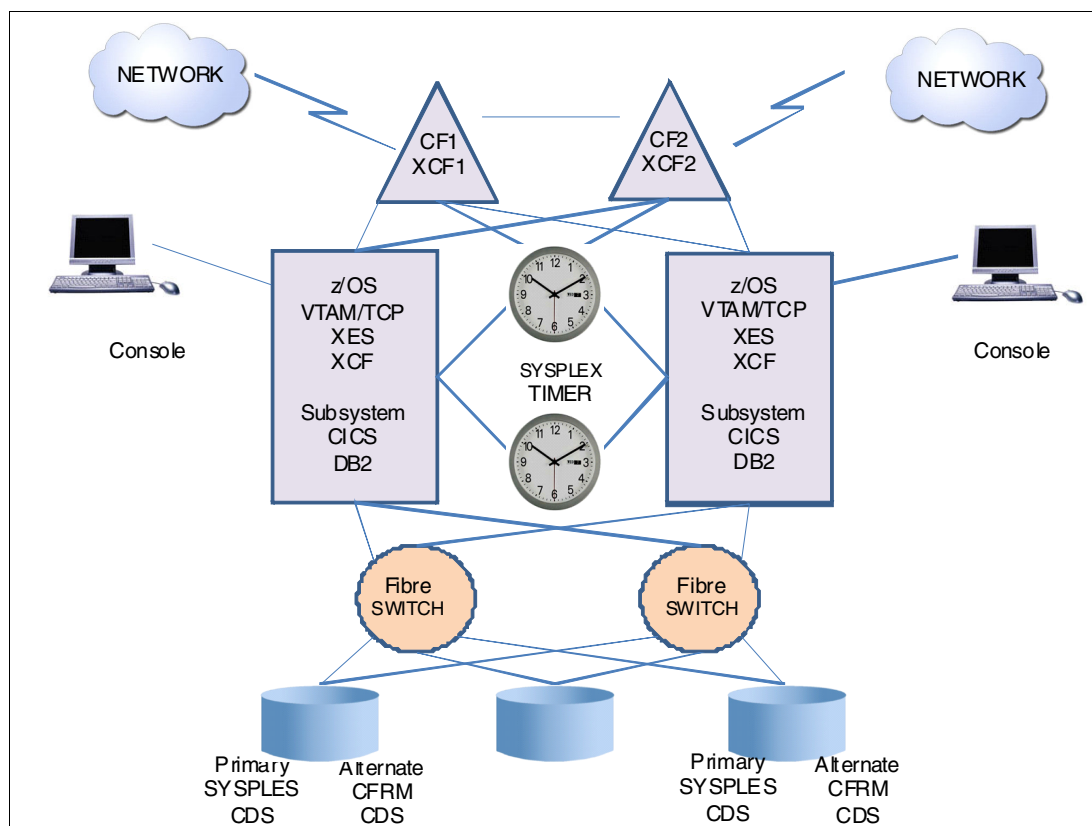


Figure 4-4 Parallel Sysplex components

The following abbreviations are used to describe parallel sysplex components:

- CF** A coupling facility (CF) is a logical partition on the System z processors. The coupling facility allows software on different systems in the sysplex to share data. The coupling facility is the hardware element that provides high-speed caching, list processing, and locking functions in a sysplex. It has no disk storage attached to it. The Coupling Facility must have connectivity to all the systems sharing the data through coupling facility channels.
- CFCC** Coupling Facility Control Code is the specialized OS which loaded from the System z Hardware. Coupling Facility Control Code stands for microcode support system, it can run on CP or ICF. ICF is an integrated coupling facility. Different levels of the coupling facility can be concurrently running in the same sysplex as long as the coupling facility LPs are running on different CPCs. CF LPs running on the same CPC share the same coupling facility control code EC level. A single CPC cannot support multiple coupling facility levels.

ICF	Integrated Coupling Facility, a designated processor to be used specifically for coupling facilities. One advantage of using ICF processors instead of CPs for Coupling Facility images is that software licenses are not charged for ICF processors. ICFs are processing units dedicated to process the CF Control Code (CFCC) on a Coupling Facility image, which is always running on a logical partition.
STP	The presence of the timers is due to a long standing requirement for accurate time and date information in data processing. The Server Time Protocol feature is designed to provide the capability for multiple servers and Coupling Facilities to maintain time synchronization with each other, without requiring a Sysplex Timer.
Structures	<p>Cache: A cache structure supplies a mechanism called buffer invalidation to ensure consistency of cached data. CF cache structures provide a level of storage hierarchy between local memory and DASD cache. They are also used as a system buffer pool for VSAM RLS data when that data is modified on other systems. Each CF cache structure is contained in a single CF. You might have multiple CFs and multiple CF cache structures.</p> <p>Lock: Lock structures supply shared and exclusive locking capability for serialization of shared resources down to a very small unit of data. The CF lock structure contains two parts: a lock entry table and a list of update locks.</p> <p>List: A CF list structure consists of a set of lists and an optional lock table of exclusive locks that you can use to serialize the use of lists, list entries, or other resources in the CF list structure. The CF can support multiple occurrences of the CF list structure at the same time. However, a CF list structure found in one CF cannot be seen by another CF.</p>

4.2.2 DB2 sysplex

One of the cornerstones of DB2's ability to support highly available applications is its support of Parallel Sysplex and sysplex data sharing.

Sysplex data sharing is based on the coupling facility. The coupling facility contains three major structures, as described in 4.2.1, "Parallel Sysplex technology", that are utilized by DB2, as shown in Figure 4-5. Each DB2 Data Sharing Group uses one Lock structure, one List structure and several cache structures (one for each Group Buffer Pool (GBP)):

- ▶ **Group Buffer Pools:** The cache structures are used as group buffer pools (GBPs), caching shared data pages for the members. These are cache structures contain data and index pages for DB2 objects that are being accessed across the data sharing group.
- ▶ **Lock structure:** The lock structure protects shared DB2 resources (such as table spaces and pages) and enables concurrent access to those resources. The lock structure ensures that the same data is not updated by multiple members at the same time.
- ▶ **Shared Communications Area (SCA):** The list structure contains the DB2 shared communications area (SCA). Each member uses the SCA to pass control information to the rest of the members in the group. The SCA contains all database exception status conditions and other information that is necessary for recovery of the group.

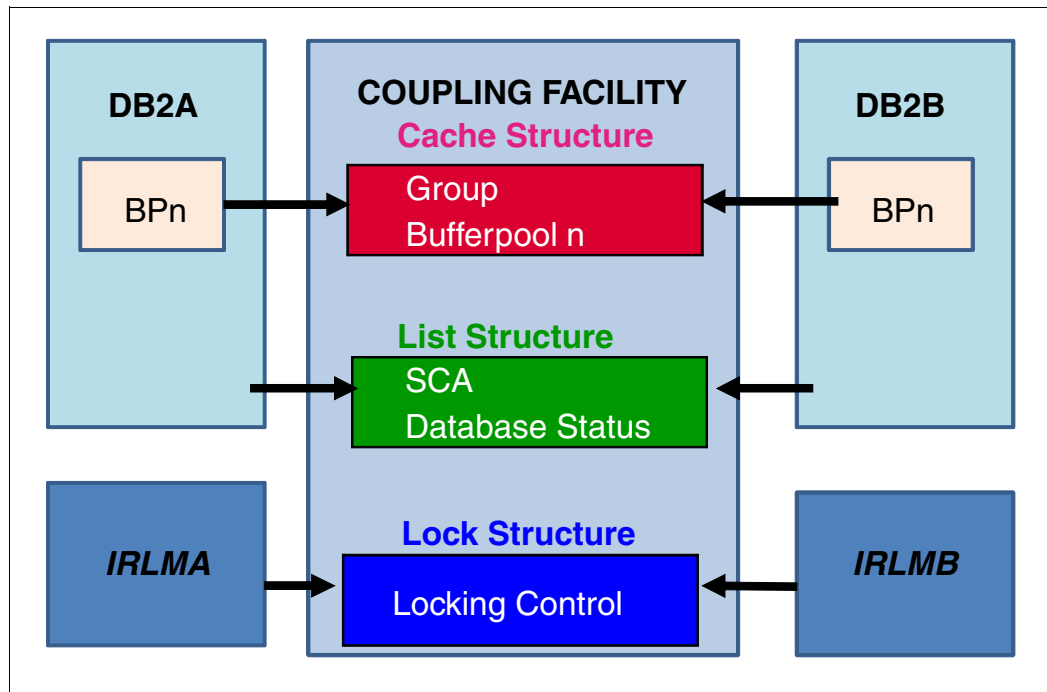


Figure 4-5 DB2's usage of the Coupling Facility

Data sharing provides the following availability features that are not provided when running a single DB2 subsystem:

- ▶ Multiple DB2 subsystems, called members of a data sharing group, that can provide failover when one of the subsystems is unavailable because of an unplanned outage
- ▶ Distribution of your workload over multiple members to reduce the CPU workload on any one member, without having to maintain multiple copies of the data
- ▶ Application of maintenance to one of the members without an outage to your production applications, which can continue to access DB2 data from the other members
- ▶ Migration to a new version of DB2 without having to bring the whole data sharing group down.

Parallel sysplex and DB2 data-sharing enables multiple nodes to work together. In this solution, there is only a single copy of the data, which is the SPOF. To remain highly availability, a second or third copy of the data will be required, which can be achieved in a coordinated manner using GDPS.

4.2.3 GDPS Metro and z/OS Global Mirror (z/OS Metro/Global Mirror)

Geographically Dispersed Parallel Sysplex (GDPS) and IBM Metro z/OS Global Mirror is a three-site solution that provides continuous availability (CA) across two sites within metropolitan distances and DR to a third site at virtually unlimited distances. It is based on a multi-target mirroring technology that combines Peer to Peer Remote Copy (PPRC) and extended Remote Copy (XRC), also known as z/OS Global Mirror on IBM storage subsystems. Figure 4-6 shows the architecture for this solution. Site 1 is the primary site, site 2 is the secondary site, and site 3 is the DR site.

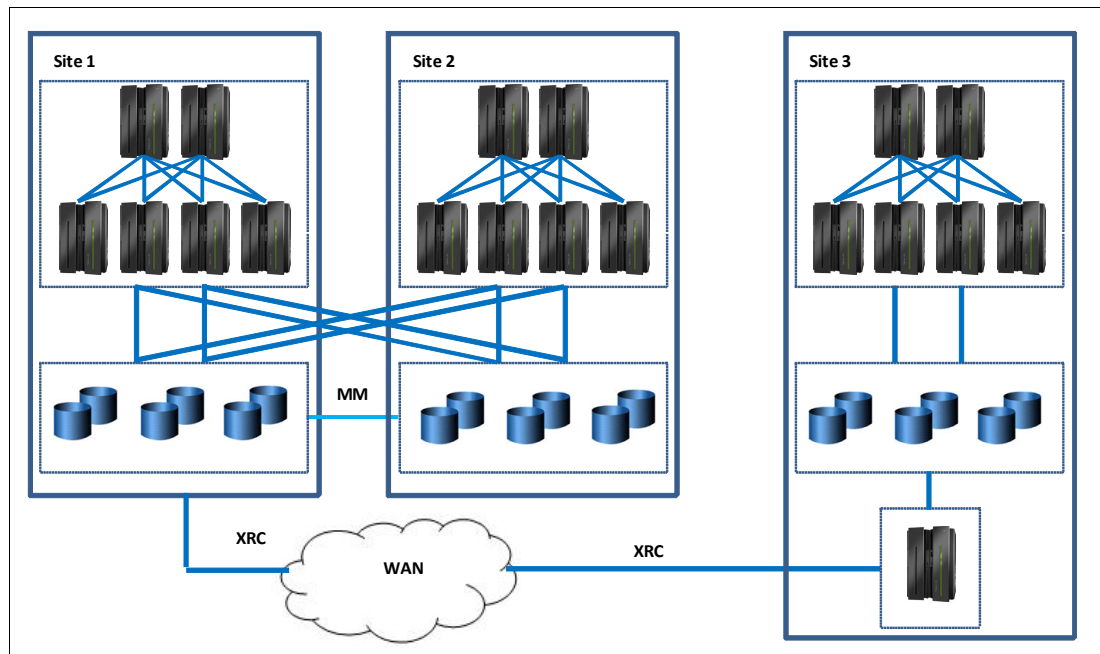


Figure 4-6 Three-site solution

GDPS/PPRC

GDPS/PPRC is a near-CA or DR solution across two sites separated by metropolitan distances. The solution is based on the IBM PPRC synchronous disk mirroring technology. GDPS/PPRC supports both planned and unplanned situations, helping to maximize application availability and provide business continuity. In particular, a GDPS/PPRC solution can deliver the following capabilities:

- ▶ Near-continuous availability solution
- ▶ Disaster Recovery (DR) solution across metropolitan distances
- ▶ Recovery Time Objective (RTO) less than an hour
- ▶ Recovery Point Objective (RPO) of zero

GDPS/PPRC delivers a powerful function known as “HyperSwap.” HyperSwap provides the ability to non-disruptively swap from using the primary volume of a mirrored pair to using what had been the secondary volume. Prior to the availability of HyperSwap, an IPL was required on every system if you wanted to switch and run from the secondary volumes, meaning that it was not possible to maintain application availability across a switch from primary to secondary volumes.

With HyperSwap, such a move can be accomplished without an IPL and with just a brief hold on application I/O. The HyperSwap function is designed to be completely controlled by automation, thus allowing all aspects of the site switch to be controlled via GDPS.

GDPS/XRC

GDPS/XRC is a DR solution that spans across two sites and is separated by virtually unlimited distance between sites. The solution is based on the IBM XRC asynchronous disk mirroring technology (also branded by IBM as z/OS Global Mirror).

Extended Remote Copy (XRC), rebranded to IBM System Storage z/OS Global Mirror, is a combined hardware and software asynchronous remote copy solution. Consistency of the data is maintained via the Consistency Group function within the z/OS System Data Mover (SDM).

Because of the asynchronous nature of XRC, it is possible to have the secondary disk at greater distances than would be acceptable for PPRC. Channel extender technology can be used to place the secondary disk up to thousands of kilometers away. Because XRC is asynchronous, the impact it has on response times is minimal, and is independent of the distance between the primary and secondary volumes.

GDPS/XRC combines the benefits of GDPS with the extended distance capabilities of XRC. It includes automation to manage remote copy pairs and automates the process of recovering the production environment with limited manual intervention, including invocation of capacity backup upgrade (CBU), thus providing significant value in reducing the duration of the recovery window and requiring less operator interaction.

Note: Capacity Backup Upgrade (CBU) for System z processors provides reserved emergency backup server capacity that can be activated in lieu of capacity that is lost as a result of an unplanned event elsewhere. CBU helps you to recover by adding reserved capacity on a designated System z system.

4.2.4 DB2 Q-replication for HA

The IBM Q Replication component of the IBM InfoSphere Data Replication product is an IBM strategic technology for deploying DB2 for z/OS active-active configurations across unlimited distance for continuous business availability throughout both planned and unplanned outages. Q Replication is a high performance log capture/ transaction-replay replication technology that uses IBM WebSphere MQ message queues to transmit and stage data between source and target database subsystems. It replicates DB2 transactions asynchronously, not impacting application response time. Q Replication is used to maintain a near real time hot failover site, where applications can be switched during planned or unplanned component or system outages.

Q replication is split into two distinct pieces:

- ▶ Q capture program or engine
- ▶ Q apply program or engine

Q capture

The Q capture program reads the DB2 logs or changes to the source table or tables that you want to replicate. These changes are then put into MQ messages and sent across the MQ infrastructure to the system where the target table resides. There they are read and applied to the target table by the Q apply program.

The Q capture program is quite flexible in terms of what can be included or excluded from the data sent to the target and even the rate at which data is sent can be modified if required. By the nature of the method of Q replication, the replication of data is an asynchronous process. Even so, an RPO of a few seconds is possible even in high update environments

Q apply

The Q apply program takes MQ messages from a receive queue, or queues and then applies the changes held within the message to the target tables. The Q apply program is designed in such a way to use parallelism to keep up with updates to multiple targets while maintaining any referential integrity constraints between related target tables.

Both the Q capture and Q apply programs have mechanisms to track what has been read from the logs and sent to the target site, and what has been read from the receive queues and applied to the target tables, including any dependencies between updates. This in turn provides data consistency and allows for restart of both the capture and apply programs, if this is required or in case of failures.

This is a resilient and software-based solution that can guarantee a shorter recovery time as it replicates only the necessary subset of data.

For more information about DB2 and Q-replication, see these publications:

- ▶ *InfoSphere Data Replication for DB2 for z/OS and WebSphere Message Queue for z/OS: Performance Lessons*, REDP-4947
- ▶ *GDPS Family: An Introduction to Concepts and Capabilities*, SG24-6374

Also see the Replication and Event Publishing section of the IBM Information Management Software for z/OS Solutions Information Center.

4.3 z/OS and Linux scenarios

The previous scenarios described high availability situations first on System z with the Linux operating system, then on System z with z/OS. In this section, we discuss a scenario with both Linux and z/OS running on System z as shown in Figure 4-7. The goal in this case is to provide a high availability solution for application systems that span the distributed and z/OS platforms.

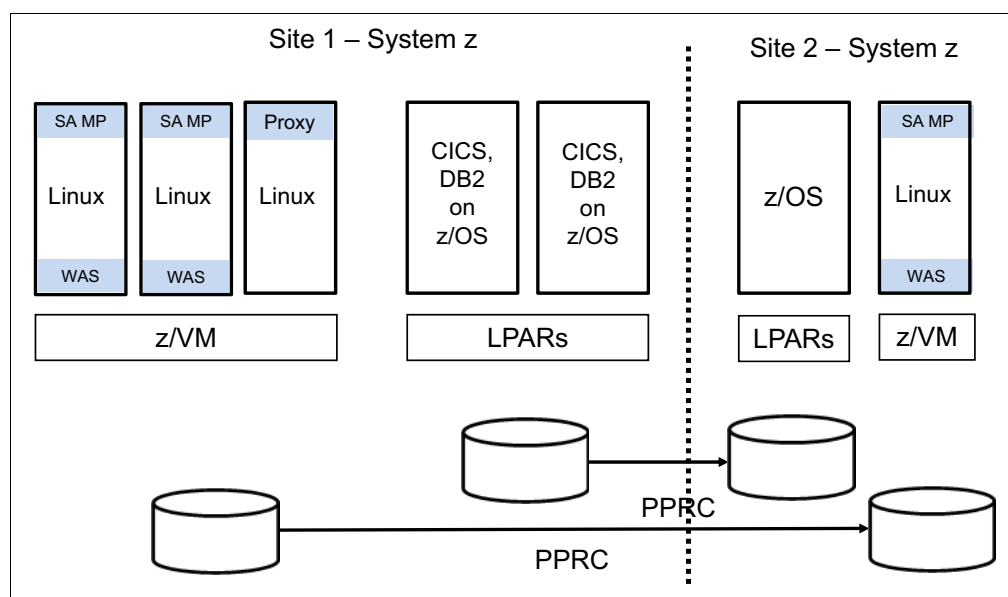


Figure 4-7 z/OS and Linux integrated configuration

In this scenario, Linux on System z is implemented as guests running under z/VM (most common), rather than native Linux on System z. In this application example, the WebSphere Application Server runs on Linux on System z and CICS and DB2 are running on z/OS.

When critical applications have a multi-tiered architecture, there is a need to provide a coordinated high availability and disaster recovery solution for both z/OS and Linux on System z. The GDPS/PPRC function that provides this capability is called Multiplatform Resiliency for System z, and it can be implemented as long as the disks being used by z/VM and Linux are CKD disks.

GDPS/PPRC provides this capability with a function called Multiplatform Resiliency for System z (also known as xDR). In this solution, GDPS/PPRC communicates and coordinates with Tivoli System Automation for MultiPlatforms (SA MP) running on Linux on System z.

In a GDPS xDR-managed z/VM system, you must configure a special Linux guest, which is known as the *proxy* guest. The proxy guest is a guest that is dedicated to providing communication and coordination with the GDPS/PPRC controlling system, and must run SA MP.

The proxy guest serves as the middleman for GDPS. It communicates commands from GDPS to z/VM, monitors the z/VM environment, and communicates status information and failure information such as a HyperSwap trigger affecting z/VM disk back to the GDPS/PPRC controlling system. GDPS/PPRC uses SA MP to pass commands to z/VM and Linux guests.

In this scenario, GDPS provides high availability to z/OS as described in section 4.2, “z/OS database scenario” on page 76.

System and hardware management capabilities similar to those available for z/OS systems are also available for z/VM systems. GDPS xDR can perform a graceful shutdown of z/VM and its guests and perform hardware actions such as LOAD and RESET against the z/VM system's partition.

This publication provides a high level introduction into the high availability capabilities of GDPS. For more detailed information, see the IBM Redbooks publication, *GDPS Family: An Introduction to Concepts and Capabilities*, SG24-6374.



Practical applications

In this chapter, we discuss an approach for creating a highly available DB2 HADR solution that covers all critical components of the database. We are using the Tivoli System Automation for Multiplatforms (SA MP) as the tool to set up this level of high availability cluster environment.

This chapter covers the following topics:

- ▶ DB2 High Availability Disaster Recovery (HADR)
- ▶ IBM Tivoli System Automation for Multiplatforms

5.1 DB2 High Availability Disaster Recovery (HADR)

The DB2 High Availability Disaster Recovery feature (DB2 HADR) is a database log replication feature that provides a high availability solution for various failure scenarios, such as the failure of an individual system or even a whole site. HADR continually replicates data changes from a primary source database to a target, or standby database. This protects against data loss. On a failure of the primary source database, the standby database becomes the new primary database.

5.1.1 DB2 HADR setup

The basic setup for DB2 HADR is displayed in Figure 5-1. It consists of two machines running Linux under z/VM in our example.

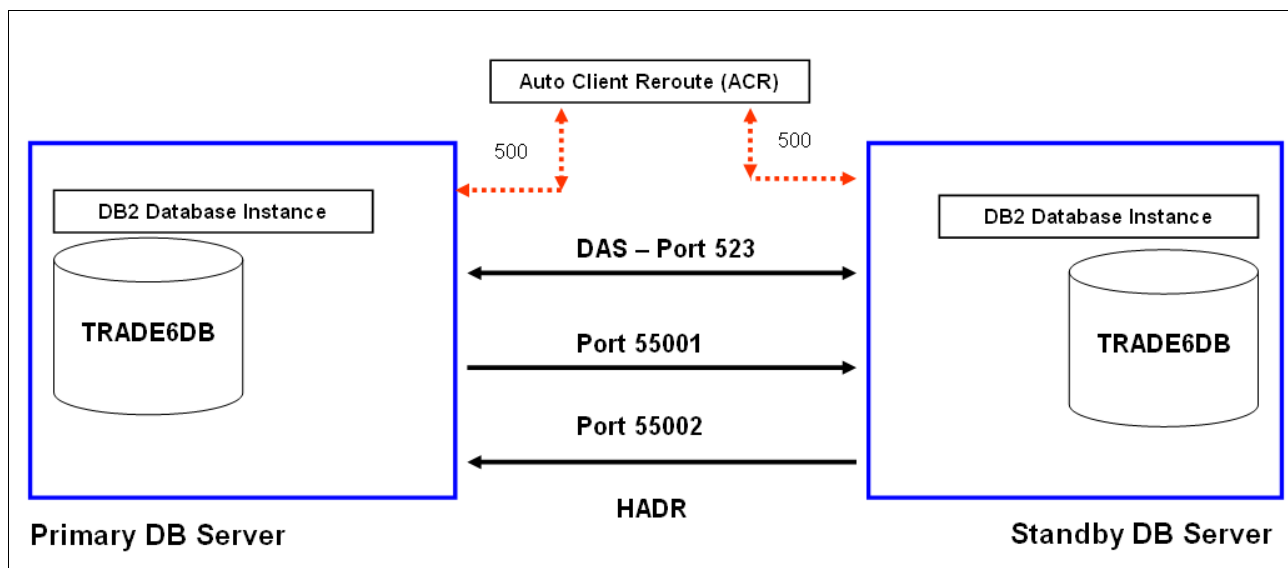


Figure 5-1 Basic DB2 HADR setup

5.1.2 DB2 HADR prerequisites

To achieve superior performance with high availability disaster recovery (HADR), you need to ensure that your cluster nodes meet the following requirements for hardware, operating systems, and for the DB2 database system:

- ▶ The operating system on the primary and standby databases should be the same version, including patches. It is also better to have the same hardware and architecture as the cluster nodes.
- ▶ A TCP/IP interface must be available between the HADR host machines, and a high-speed, high-capacity network is recommended.
- ▶ The DB2 database software for both the primary and standby databases must have the same architecture bit size (32 or 64 bit).

Example 5-1 shows the Linux and DB2 version on the primary node in our environment.

Example 5-1 Primary Site Linux and DB2 Version

```
db2inst1@lnsudb1:~> cat /etc/issue
Welcome to SUSE Linux Enterprise Server 11 SP3 (s390x) - Kernel \r (\l).

db2inst1@lnsudb1:~> uname -a
Linux lnsudb1 3.0.76-0.11-default #1 SMP Fri Jun 14 08:21:43 UTC 2013 (ccab990)
s390x s390x s390x GNU/Linux

db2inst1@lnsudb1:~> db2level
DB21085I This instance or install (instance name, where applicable:
"db2inst1") uses "64" bits and DB2 code release "SQL10053" with level
identifier "0604010E".
Informational tokens are "DB2 v10.5.0.3", "s140203", "IP23553", and Fix Pack
"3".
Product is installed at "/opt/ibm/db2/V10.5".
```

In Example 5-2, we issued the same command on the standby node to make sure that the configurations were the same as the primary node.

Example 5-2 Standby Site Linux and DB2 Version

```
db2inst1@lnsudb3:~> cat /etc/issue
Welcome to SUSE Linux Enterprise Server 11 SP3 (s390x) - Kernel \r (\l).

db2inst1@lnsudb3:~> uname -a
Linux lnsudb3 3.0.76-0.11-default #1 SMP Fri Jun 14 08:21:43 UTC 2013 (ccab990)
s390x s390x s390x GNU/Linux

db2inst1@lnsudb3:~> db2level
DB21085I This instance or install (instance name, where applicable:
"db2inst1") uses "64" bits and DB2 code release "SQL10053" with level
identifier "0604010E".
Informational tokens are "DB2 v10.5.0.3", "s140203", "IP23553", and Fix Pack
"3".
Product is installed at "/opt/ibm/db2/V10.5".
```

We created a sample database layout (TRADE6DB) and other database objects on just one database which initially will act as the primary node. When this is transferred to the standby database, it will make the database and its objects consistent between the primary and standby nodes.

5.1.3 Setting up the DB2 registry profile

The DB2COMM registry variable allows you to set communication protocols for the current DB2 instance. If the DB2COMM registry variable is undefined or set to null, no protocol connection managers are started. Both the primary and secondary server must have this parameter set. All HADR registry variables changes require a DB2 instance shutdown and restart to take effect. Prior to DB2 V10.1, HADR database configuration parameters require database shutdown and restart to take effect. For HADR to work, the DB2COMM variable needs to be set to TCPIP, as shown in Example 5-3 on page 88.

Example 5-3 DB2 Registry profile

```
db2inst1@lnsudb1:~> db2set DB2COMM=tcpip
```

```
db2inst1@lnsudb1:~> db2set -all
[i] DB2COMM=TCPIP
[i] DB2AUTOSTART=NO
[g] DB2SYSTEM=lnsudb1
[g] DB2INSTDEF=db2inst1
[g] DB2ADMINSERVER=dasusr1
```

5.1.4 Linux ports configuration

For setting up HADR for DB2, separate sets of HADR ports have to be defined in the `/etc/services` file in Linux. The relevant part is displayed in Example 5-4. In our environment, the primary and standby port numbers are 55001 and 55002. These ports must be identified as TCP/IP services.

Example 5-4 Primary Server services file

```
DB2_db2inst1      60004/tcp
DB2_db2inst1_1    60005/tcp
DB2_db2inst1_2    60006/tcp
DB2_db2inst1_END  60007/tcp
DB2_HADR_1        55001/tcp
DB2_HADR_2        55002/tcp
```

5.1.5 Database archive setup

Both the primary database and the standby databases should be configured for log archiving, as any of these systems could be promoted to become the primary, at which point it will be required to archive its log files. Only the current primary database can perform log archiving. In the event of a takeover, the standby database becomes the new primary database and any logs archived from that point on are saved to the original standby database's archiving location. Example 5-5 details the commands needed to configure log archiving.

Example 5-5 Command Line for Archive log

```
db2inst1@lnsudb1:~> db2 update db cfg for TRADE6DB using logretain on
db2inst1@lnsudb1:~> db2 update db cfg for TRADE6DB using LOGINDEXBUILD on
```

If IBM Data Studio is used for configuring DB2 HADR, then before anything else, the tool would automatically warn and lead us to configure archive logging, as shown in Figure 5-2. Further dialogs would request the user to specify a directory location to save the archive logs.

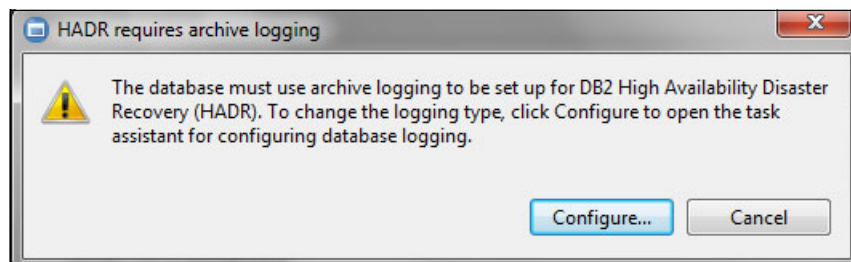


Figure 5-2 IBM Data Studio HADR setup

5.1.6 Backup of the primary database

Before the primary and standby server are configured for HADR, a full back up of the primary database has to be taken. This can be either done using the command line tools (as in Example 5-6) or it can be done using IBM Data Studio.

Example 5-6 Primary database backup

```
db2inst1@lnsldb1:~> db2 backup db trade6db to /db2_back
Backup successful. The timestamp for this backup is : 20140617170937
```

Now that the backup of the database is completed, it is time to transfer the database backup to the standby system and restore it. We used the secure copy (SCP) protocol to transfer the database between the nodes. The command used is displayed in Example 5-7. Other protocols, such as FTP, can also be used, but care needs to be taken that the database does not become corrupted or that the integrity becomes compromised during the transfer.

Example 5-7 Transferring backup database to standby node

```
lnsldb1:/db2_back # scp TRADE6DB.0.db2inst1.DBPART000.20140617170937.001
9.12.7.90:/db2_back/.
```

To restore the database, use the DB2 command shown in Example 5-8. Ensure that the database restoration completes successfully.

Example 5-8 Recovery of database on standby system

```
db2inst1@lnsldb3:~> db2 restore db trade6db
DB20000I The RESTORE DATABASE command completed successfully.
```

5.1.7 Setting up the DB2 database configuration

For setting up HADR, we need to set up DB2 configuration parameters in both primary and standby nodes. It is always better to decide on the HADR configuration parameters (such as IP address and ports) as much as you can before initializing HADR. This minimizes the activities involved in updating the configuration after initialization. Changing the configuration after setup may require the DB2 instance or database to be shut down and restarted. Example 5-9 shows our DB2 environment before setting up HADR.

Example 5-9 Before HADR setup

```
db2 get db cfg for trade6db

HADR database role                                = STANDARD
HADR local host name                             (HADR_LOCAL_HOST) =
HADR local service name                         (HADR_LOCAL_SVC) =
HADR remote host name                           (HADR_REMOTE_HOST) =
HADR remote service name                       (HADR_REMOTE_SVC) =
HADR instance name of remote server             (HADR_REMOTE_INST) =
HADR timeout value                             (HADR_TIMEOUT) =
HADR target list                                (HADR_TARGET_LIST) =
HADR log write synchronization mode              (HADR_SYNCMODE) =
HADR spool log data limit (4KB)                 (HADR_SPOOL_LIMIT) = 0
HADR log replay delay (seconds)                 (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)            (HADR_PEER_WINDOW) = 0
```

Issue the set of commands on both the primary (as shown in Example 5-10) and standby (as shown in Example 5-11) respectively. See Figure 5-1 on page 86 for IP address and port details of the local node and remote node.

Example 5-10 Primary node configuration parameters

```
db2 update db cfg for trade6db using HADR_LOCAL_HOST lnsudb1
db2 update db cfg for trade6db using HADR_LOCAL_SVC 55001
db2 update db cfg for trade6db using HADR_REMOTE_HOST lnsudb3
db2 update db cfg for trade6db using HADR_REMOTE_SVC 55002
db2 update db cfg for trade6db using HADR_TARGET_LIST lnsudb3:55002
```

Example 5-11 Standby node Configuration parameters setup commands

```
db2 update db cfg for trade6db using HADR_LOCAL_HOST lnsudb3
db2 update db cfg for trade6db using HADR_LOCAL_SVC 55002
db2 update db cfg for trade6db using HADR_REMOTE_HOST lnsudb1
db2 update db cfg for trade6db using HADR_REMOTE_SVC 55001
db2 update db cfg for trade6db using HADR_TARGET_LIST lnsudb1:55001
```

Verify the DB2 configuration parameters after all the parameters are applied onto the primary and standby nodes. Example 5-12 and Example 5-13 display the resulting configurations on both nodes.

Example 5-12 Primary node updated configuration parameters

```
db2inst1@lnsudb1:~> db2 get db cfg for trade6db
HADR database role                      = STANDARD
HADR local host name                    (HADR_LOCAL_HOST) = lnsudb1
HADR local service name                  (HADR_LOCAL_SVC) = 55001
HADR remote host name                    (HADR_REMOTE_HOST) = lnsudb3
HADR remote service name                  (HADR_REMOTE_SVC) = 55002
HADR instance name of remote server      (HADR_REMOTE_INST) = db2inst1
HADR timeout value                       (HADR_TIMEOUT)    = 120
HADR target list                         (HADR_TARGET_LIST) = lnsudb3:55002
HADR log write synchronization mode       (HADR_SYNCMODE)   = NEARSYNC
HADR spool log data limit (4KB)           (HADR_SPOOL_LIMIT) = 0
HADR log replay delay (seconds)           (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)       (HADR_PEER_WINDOW) = 0
```

Example 5-13 Secondary node configuration parameters

```
db2inst1@lnsudb3:~> db2 get db cfg for trade6db
HADR database role                      = STANDARD
HADR local host name                    (HADR_LOCAL_HOST) = lnsudb3
HADR local service name                  (HADR_LOCAL_SVC) = 55002
HADR remote host name                    (HADR_REMOTE_HOST) = lnsudb1
HADR remote service name                  (HADR_REMOTE_SVC) = 55001
HADR instance name of remote server      (HADR_REMOTE_INST) = db2inst1
HADR timeout value                       (HADR_TIMEOUT)    = 120
HADR target list                         (HADR_TARGET_LIST) = lnsudb1:55001
HADR log write synchronization mode       (HADR_SYNCMODE)   = NEARSYNC
HADR spool log data limit (4KB)           (HADR_SPOOL_LIMIT) = AUTOMATIC(0)
HADR log replay delay (seconds)           (HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)       (HADR_PEER_WINDOW) = 0
```

5.1.8 Starting HADR

Now that we have completed the setup of the configuration parameters, we are ready to test start the HADR clustering between the nodes. Always start the standby database first and then the primary database. Example 5-14 is the command we ran to start the standby HADR database.

Example 5-14 Standby Node HADR Initiation

```
db2inst1@lnsldb3:~> db2 start hadr on db TRADE6DB as standby
SQL1766W The command completed successfully. However, LOGINDEXBUILD was not
enabled before HADR was started.
db2inst1@lnsldb3:~>
```

Now that HADR is started on the standby node, we are able to check the configuration parameters for any change in the standby HADR database role, using the command in Example 5-15. If HADR is started without any issues, then the HADR database role will be changed to standby.

Example 5-15 After starting HADR: Standby node configuration

```
db2inst1@lnsldb3:~> db2 get db cfg for trade6db
```

HADR database role	= STANDBY
HADR local host name	(HADR_LOCAL_HOST) = lnsldb3
HADR local service name	(HADR_LOCAL_SVC) = 55002
HADR remote host name	(HADR_REMOTE_HOST) = lnsdb1
HADR remote service name	(HADR_REMOTE_SVC) = 55001
HADR instance name of remote server	(HADR_REMOTE_INST) = db2inst1
HADR timeout value	(HADR_TIMEOUT) = 120
HADR target list	(HADR_TARGET_LIST) = lnsdb1:55001
HADR log write synchronization mode	(HADR_SYNCMODE) = NEARSYNC
HADR spool log data limit (4KB)	(HADR_SPOOL_LIMIT) = AUTOMATIC(25600)
HADR log replay delay (seconds)	(HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)	(HADR_PEER_WINDOW) = 0

We can go ahead and start the HADR on the primary node, as in Example 5-16. It is a good practice to activate the database before starting the HADR on the primary node.

Example 5-16 Primary node: HADR initiation

```
db2inst1@lnsdb1:~> db2 activate db TRADE6DB
DB20000I The ACTIVATE DATABASE command completed successfully.
db2inst1@lnsdb1:~>

db2inst1@lnsdb1:~> db2 start hadr on db TRADE6DB as primary
DB20000I The START HADR ON DATABASE command completed successfully.
db2inst1@lnsdb1:~>
```

After starting, we checked the configuration parameter, using the command shown in Example 5-17.

Example 5-17 After starting HADR: Primary node configuration

```
db2inst1@lnsldb3:~> db2 get db cfg for trade6db
HADR database role = PRIMARY
HADR local host name (HADR_LOCAL_HOST) = lnsdb1
```

HADR local service name	(HADR_LOCAL_SVC) = 55001
HADR remote host name	(HADR_REMOTE_HOST) = lnsudb3
HADR remote service name	(HADR_REMOTE_SVC) = 55002
HADR instance name of remote server	(HADR_REMOTE_INST) = db2inst1
HADR timeout value	(HADR_TIMEOUT) = 120
HADR target list	(HADR_TARGET_LIST) = lnsudb3:55002
HADR log write synchronization mode	(HADR_SYNCMODE) = NEARSYNC
HADR spool log data limit (4KB)	(HADR_SPOOL_LIMIT) = 0
HADR log replay delay (seconds)	(HADR_REPLAY_DELAY) = 0
HADR peer window duration (seconds)	(HADR_PEER_WINDOW) = 0

As the HADR ports are not activated, we can check whether the ports are listening as expected using network statistic tools in Linux. In our case, the HADR ports 55001 and 55002 had to be in Listen mode, which is confirmed by Example 5-18.

Example 5-18 Netstat command to check on the HADR ports

```
db2inst1@lnsudb1:~> netstat -antp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:60004          0.0.0.0:*               LISTEN      3645/db2sysc 0
tcp        0      0 0.0.0.0:5801          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:523           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:5901          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:111           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:21            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22            0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:631         0.0.0.0:*               LISTEN      -
tcp        0      0 9.12.7.86:55001       0.0.0.0:*               LISTEN      3645/db2sysc 0
tcp        0      0 127.0.0.1:25          0.0.0.0:*               LISTEN      -
tcp        0      0 9.12.7.86:60004       9.57.138.181:54528      ESTABLISHED 3645/db2sysc 0
tcp        0      0 9.12.7.86:55001       9.12.7.90:42886         ESTABLISHED 3645/db2sysc 0
tcp        0      0 9.12.7.86:60004       9.57.138.181:54640      ESTABLISHED 3645/db2sysc 0
tcp        0      52 9.12.7.86:22          9.57.138.181:54504      ESTABLISHED -
tcp        0      0 9.12.7.86:60004       9.57.138.181:54641      ESTABLISHED 3645/db2sysc 0
tcp        0      0 9.12.7.86:22          9.57.138.181:54648      ESTABLISHED -
tcp        0      0 :::111                :::*                    LISTEN      -
tcp        0      0 :::80                  :::*                    LISTEN      -
tcp        0      0 :::22                  :::*                    LISTEN      -
tcp        0      0 :::1:631               :::*                    LISTEN      -
tcp        0      0 :::1:25                :::*                    LISTEN      -
```

5.1.9 DB2 HADR takeover verification

Now that we have HADR configured and running, we tested our setup with a real life scenario of primary database disconnection. From the primary database server, we killed the **db2sysc** process, as shown in Example 5-19. Without this process, the database server would not function.

Example 5-19 Initiating a database crash in primary server

```
lnsudb1:~ # ps -ef | grep db2sys
db2inst1 3689 3687 0 Jun19 ?          00:01:44 db2sysc 0
lnsudb1:~ # kill 3687
```

Because the primary database was not reachable, we manually initiated a takeover of the database on the standby server, using the procedure shown in Example 5-20.

Example 5-20 Initiate Manual takeover of Database in standby server

```
db2inst1@lnsudb3:~> db2 takeover hadr on db trade6db
DB20000I The TAKEOVER HADR ON DATABASE command completed successfully.
db2inst1@lnsudb3:~>
```

```
db2inst1@lnsudb3:~> db2pd -hadr -db trade6db
```

```
Database Member 0 -- Database TRADE6DB -- Active -- Up 1 days 17:19:22 -- Date
2014-06-19-10.43.01.096991
```

```

      HADR_ROLE = PRIMARY
      REPLAY_TYPE = PHYSICAL
      HADR_SYNCMODE = NEARSYNC
      STANDBY_ID = 1
      LOG_STREAM_ID = 0
      HADR_STATE = PEER
      HADR_FLAGS =
      PRIMARY_MEMBER_HOST = lnsudb3
      PRIMARY_INSTANCE = db2inst1
      PRIMARY_MEMBER = 0
      STANDBY_MEMBER_HOST = lnsudb1
      STANDBY_INSTANCE = db2inst1
      STANDBY_MEMBER = 0
      HADR_CONNECT_STATUS = CONNECTED
      HADR_CONNECT_STATUS_TIME = 06/17/2014 17:29:28.342366 (1403040568)
      HEARTBEAT_INTERVAL(seconds) = 30
      HADR_TIMEOUT(seconds) = 120
      TIME_SINCE_LAST_RECV(seconds) = 17
      PEER_WAIT_LIMIT(seconds) = 0
      LOG_HADR_WAIT_CUR(seconds) = 0.000
      LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
      LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
      LOG_HADR_WAIT_COUNT = 0
      SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
      SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
      PRIMARY_LOG_FILE,PAGE,POS = S0000016.LOG, 56, 162032321
      STANDBY_LOG_FILE,PAGE,POS = S0000016.LOG, 56, 162032321
      HADR_LOG_GAP(bytes) = 0
      STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000016.LOG, 56, 162032321
      STANDBY_RECV_REPLAY_GAP(bytes) = 0
      PRIMARY_LOG_TIME = 06/19/2014 08:38:17.000000 (1403181497)
      STANDBY_LOG_TIME = 06/19/2014 08:38:17.000000 (1403181497)
      STANDBY_REPLAY_LOG_TIME = 06/19/2014 08:38:17.000000 (1403181497)
      STANDBY_RECV_BUF_SIZE(pages) = 982
      STANDBY_RECV_BUF_PERCENT = 0
      STANDBY_SPOOL_LIMIT(pages) = 0
```

```
STANDBY_SPOOL_PERCENT = NULL
PEER_WINDOW(seconds) = 0
READS_ON_STANDBY_ENABLED = N
```

5.1.10 Why we used Tivoli System Automation

As we have seen, even though DB2 HADR provides failover capability, manual intervention is required for identifying the database failure and also initiating the failover. Because of this, the Recovery Time Objective (RTO) for this failover would be high, because this requires manual intervention.

Hence, if it is a mission critical database that would require an RTO with zero or near zero turn-around time, then a high availability solution with only DB2 HADR would not be an option. DB2 HADR has to be automated with tools or scripts to avoid manual intervention and to achieve near zero Real Time Objective (RTO).

The DB2 HADR solution is used for maintaining a “hot standby,” but the actual takeover making the standby database the primary one, and potentially reintegrating the old primary database again, requires manual intervention. This manual intervention consists of specific HADR commands. Therefore, there is a need for a clustering software, which can automate the failover of resources, such as processes, applications, storage, and IP addresses. Clustering software monitors the health of the network, hardware, and software processes, detects and communicates any fault, and automatically fails over the service and associated resources to a healthy host in the cluster.

In this chapter, we use Tivoli System Automation for Multiplatforms (SA MP) as our clustering software. SA MP is tightly integrated with most of the IBM middleware products and can be used to provide efficient failovers, that is, for example, detecting the outage of the primary system and automatically issuing the specific DB2 HADR commands.

5.2 IBM Tivoli System Automation for Multiplatforms

Tivoli System Automation for Multiplatforms (sometimes referred to as TSA MP or SA MP) is a high availability cluster solution that provides several monitoring mechanisms to detect system failures and a set of rules to initiate the correct action without any user intervention. The set of rules is called a policy and this policy describes the relationships between applications or resources. This policy and the monitoring mechanisms provide SA MP with extensive up-to-date information about the system landscape so that it can restart the resource on the current node or move the whole database to another cluster node.

SA MP is part of each DB2 installation, and the High Availability Disaster Recovery setup utility, **db2haicu**, can be used to configure HADR and a respective System Automation for Multiplatforms policy.

You can install, upgrade, or uninstall SA MP using either the DB2 installer or the installSAM and uninstallSAM scripts that are included in the DB2 server installation media.

5.2.1 Prerequisites

By default, the SA MP components are installed with DB2. The following prerequisites must be satisfied before installation:

- ▶ Depending on the distributions, the following packages have to be installed to satisfy dependencies:
 - libstdc++
 - compat-lib
 - glibc
 - ksh
 - glibc-locale
- ▶ Both cluster nodes have to be installed with the same version of the tool, else the scripts used for automation purposes by SA MP may vary and could result in unpredictable outcomes.
- ▶ Tivoli System Authomation requires users to define all the cluster nodes with fully qualified node names (IP Address, long name and short name) in the `/etc/hosts` file.
- ▶ A virtual IP address is required by Tivoli System Authomation for failover purposes. It would float the IP between the primary and standby, in the event of node failures.

5.2.2 Tivoli System Authomation manual installation

After all the prerequisite RPMs are installed, we can proceed with the actual installation of SA MP using the installation script provided in the DB2 installation media.

In our environment, we have mounted the DB2 v10.1 installation media under `/opt/server` and the SA MP installation source can be found in the file system `/opt/serverdb2/linux390/tsamp/`, as shown in Example 5-21. The location of the SA MP installation files can vary according to where the media is mounted in the environment.

Example 5-21 SA MP Installation media

```
lnsddb1:/opt/server/db2/linux390/tsamp # ls -la
total 296
drwxr-xr-x 6 bin bin 4096 Feb 4 14:32 .
drwxr-xr-x 8 bin bin 4096 Feb 4 14:35 ..
drwxr-xr-x 6 bin bin 4096 Feb 4 14:31 Linux
drwxr-xr-x 2 bin bin 4096 Feb 4 14:31 README
lrwxrwxrwx 1 root root 19 Feb 4 14:31 db2cktsa -> ../install/db2cktsa
-r-xr-xr-x 1 root root 134909 Feb 4 14:31 installSAM
drwxr-xr-x 2 bin bin 4096 Feb 4 14:32 integration
drwxr-xr-x 2 bin bin 4096 Feb 4 14:32 license
-r-xr-xr-x 1 root root 84364 Feb 4 14:31 prereqSAM
-r-xr-xr-x 1 root root 46642 Feb 4 14:31 uninstallSAM
```

For installing SA MP, we need to login as root and execute the **prereqSAM** script, which does a prerequisites check on the environment. After the prerequisites are complete, initiate the **installSAM** script to install SA MP. Example 5-22 shows the output of these commands.

Example 5-22 Installing SA MP

```
lnsddb1:/opt/server/db2/linux390/tsamp # ./prereqSAM
prereqSAM: All prerequisites for the ITSAMP installation are met on operating
system
SUSE Linux Enterprise Server 11 (s390x)
```

```
VERSION = 11
PATCHLEVEL = 2

lnsudb1:/opt/server/db2/linux390/tsamp # ./installSAM
prereqSAM: All prerequisites for the ITSAMP installation are met on operating
system
SUSE Linux Enterprise Server 11 (s390x)
VERSION = 11
PATCHLEVEL = 2
...
installSAM: All packages were installed successfully.
```

Now that we have completed the SA MP installation, we can proceed with configuring it to automate DB2 HADR. This can be done using the **db2haicu** command.

5.2.3 Preparing the node for communication

Before we proceed with the **db2haicu** command, we need to prepare the nodes for communication. This requires that the DB2 configuration files (**db2nodes.cfg**) have the same longname or shortname as HADR and that the hostname command returns the same names, as shown in Example 5-23.

Example 5-23 Node name check in db2nodes.cfg

```
db2inst1@lnsudb1:~> cat ~/sqllib/db2nodes.cfg
0 lnsudb1 0

db2inst1@lnsudb3:~> cat ~/sqllib/db2nodes.cfg
0 lnsudb3 0
```

RSCT service

During the installation SA MP, Reliable Scalable Cluster Technology (RSCT) is also installed. RSCT is a set of software components that together provide a comprehensive clustering environment in Linux and other supported operating systems. SA MP uses RSCT to provide clusters with improved system availability, scalability, and ease of use and hence the RSCT service has to be started on all the cluster nodes, using the command shown in Example 5-24.

Example 5-24 Starting RSCT

```
lnsudb1:~ # /usr/sbin/rsct/install/bin/recfgct -s
0513-071 The ctcas Subsystem has been added.
0513-071 The ctrmc Subsystem has been added.
0513-059 The ctrmc Subsystem has been started. Subsystem PID is 24045.

lnsudb3:~ # /usr/sbin/rsct/install/bin/recfgct -s
0513-071 The ctcas Subsystem has been added.
0513-071 The ctrmc Subsystem has been added.
0513-059 The ctrmc Subsystem has been started. Subsystem PID is 5602.
```

The preprnode command

The **preprnode** command prepares security on the node on which the command is run so it can be defined in a cluster domain. It allows for cluster domain operations to be performed on the node and must be run before the node can join a cluster. This command has to be executed on every node that will be part of the automation domain. Example 5-25 demonstrates this. There is no specific output from the command execution.

Example 5-25 initiating preprnode

```
lnsudb1:~ # preprnode lnsudb1 lnsudb3
lnsudb1:~ #
```

```
lnsudb3:~ # preprnode lnsudb1 lnsudb3
lnsudb3:~ #
```

Note: The **preprnode** command exchanges information (such as keys) during execution. If you have a firewall enabled, then the command will fail.

5.2.4 SA MP Cluster configuration

SA MP uses the **db2haicu** script to configure DB2 HADR failover, and this script can be used either in interactive mode or silent mode, using an XML response file. With interactive mode, the script would ask questions for which the user needs to answer details about the environment (that is, virtual IP address, node names, and so on). For first time users of **db2haicu**, use **db2haicu** in interactive mode.

In our environment, we have used the **db2haicu** silent configuration using an XML configuration file. A sample script of this can be found in this path:

```
/opt/ibm/db2/V10.5/samples/ha/xml/db2ha_sample_HADR.xml
```

In Example 5-26 we have edited the sample XML file with the values relevant to our environment. The same XML file will be used on both the primary and standby nodes.

Example 5-26 db2haicu XML file for silent configuration

```
<DB2Cluster xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="db2ha.xsd" clusterManagerName="TSA" version="1.0">

<ClusterDomain domainName="itsodm1">
  <Quorum quorumDeviceProtocol="network" quorumDeviceName="9.12.4.1"/>

  <PhysicalNetwork physicalNetworkName="db2_public_network_0"
physicalNetworkProtocol="ip">
    <Interface interfaceName="eth0" clusterNodeName="lnsudb3">
      <IPAddress baseAddress="9.12.7.90" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
    </Interface>
    <Interface interfaceName="eth0" clusterNodeName="lnsudb1">
      <IPAddress baseAddress="9.12.7.86" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
    </Interface>
  </PhysicalNetwork>

  <ClusterNode clusterNodeName="lnsudb1"/>
```

```

    <ClusterNode clusterNodeName="lnsldb3"/>
</ClusterDomain>

<FailoverPolicy>
    <HADRFailover></HADRFailover>
</FailoverPolicy>

<DB2PartitionSet>
    <DB2Partition dbpartitionnum="0" instanceName="db2inst1">
    </DB2Partition>
</DB2PartitionSet>

<HADRDBSet>
    <HADRDB databaseName="TRADE6DB" localInstance="db2inst1"
remoteInstance="db2inst1" localHost="lnsldb1" remoteHost="lnsldb3" />
    <VirtualIPAddress baseAddress="9.12.7.97" subnetMask="255.255.252.0"
networkName="db2_public_network_0"/>
</HADRDBSet>
</DB2Cluster>

```

Note that in the XML file, we have defined the virtual IP address that would be used by SA MP to setup VIPA in the primary node and be used as a floating IP address between primary and standby in case of failover. Also, the quorum device address must be provided in the XML configuration file. SA MP uses the quorum device IP address as the tie-breaker in case the network connectivity is lost between the nodes. In our environment, the network Gateway IP address would act as a quorum device.

Now that we have completed modifying the XML file used by **db2haicu** for a silent configuration, we can initiate the **db2haicu** command on the standby node. Execute **db2haicu** on the standby node first and then on the primary node. Example 5-27 shows the output of the command run on the standby node.

Example 5-27 db2haicu on the standby node

```

db2inst1@lnsldb1:~> db2haicu -f db2ha_itso_HADR.xml
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is 'db2inst1'. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...

Configuring quorum device for domain 'itsodm1' ...

Configuring quorum device for domain 'itsodm1' was successful.

Network adapter 'eth0' on node 'lnsldb3' is already defined in network 'db2_public_network_0' and cannot be added to another network until it is removed from its current network.


```

Network adapter 'eth0' on node 'lnsudb1' is already defined in network
'db2_public_network_0' and cannot be added to another network until it is removed
from its current network.
Adding DB2 database partition '0' to the cluster ...
Adding DB2 database partition '0' to the cluster was successful.
Adding HADR database 'TRADE6DB' to the domain ...
Adding HADR database 'TRADE6DB' to the domain was successful.
All cluster configurations have been completed successfully. db2haicu exiting ...
db2inst1@lnsudb1:~>

```

```

db2inst1@lnsudb3:~> db2haicu -f db2ha_itso_HADR.xml
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is 'db2inst1'. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...

```

Creating domain 'itsodom1' in the cluster ...
Creating domain 'itsodom1' in the cluster was successful.
Configuring quorum device for domain 'itsodom1' ...
Configuring quorum device for domain 'itsodom1' was successful.
Adding network interface card 'eth0' on cluster node 'lnsudb3' to the network
'db2_public_network_0' ...
Adding network interface card 'eth0' on cluster node 'lnsudb3' to the network
'db2_public_network_0' was successful.
Adding network interface card 'eth0' on cluster node 'lnsudb1' to the network
'db2_public_network_0' ...
Adding network interface card 'eth0' on cluster node 'lnsudb1' to the network
'db2_public_network_0' was successful.
Adding DB2 database partition '0' to the cluster ...
Adding DB2 database partition '0' to the cluster was successful.
HADR database 'TRADE6DB' has been determined to be valid for high availability.
However, the database cannot be added to the cluster from this node because
db2haicu detected this node is the standby for HADR database 'TRADE6DB'. Run
db2haicu on the primary for HADR database 'TRADE6DB' to configure the database for
automated failover.
All cluster configurations have been completed successfully. db2haicu exiting ...
db2inst1@lnsudb3:~>

```

This process brings us almost to the state of a working cluster. The HADR database pair is not added to the cluster yet because the database on the current cluster node is in the HADR standby role. We can initiate the command on the primary node to include both the nodes in the cluster, as shown in Example 5-28 on page 100.

Important: The same XML file we created has to be used with both the primary and the standby nodes. The **db2haicu** command would use the XML definitions on to the nodes.

Example 5-28 Executing db2haicu on the primary node

```
db2inst1@lnsddb3:~> db2haicu -f db2ha_itso_HADR.xml
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
```

You can find detailed diagnostic information in the DB2 server diagnostic log file called `db2diag.log`. Also, you can use the utility called `db2pd` to query the status of the cluster domains you create.

For more information about configuring your clustered environment using `db2haicu`, see the topic called 'DB2 High Availability Instance Configuration Utility (`db2haicu`)' in the DB2 Information Center.

`db2haicu` determined the current DB2 database manager instance is 'db2inst1'. The cluster configuration that follows will apply to this instance.

`db2haicu` is collecting information on your current setup. This step may take some time as `db2haicu` will need to activate all databases for the instance to discover all paths ...

Creating domain 'itsodom1' in the cluster ...

Creating domain 'itsodom1' in the cluster was successful.

Configuring quorum device for domain 'itsodom1' ...

Configuring quorum device for domain 'itsodom1' was successful.

Adding network interface card 'eth0' on cluster node 'lnsddb3' to the network 'db2_public_network_0' ...

Adding network interface card 'eth0' on cluster node 'lnsddb3' to the network 'db2_public_network_0' was successful.

Adding network interface card 'eth0' on cluster node 'lnsddb1' to the network 'db2_public_network_0' ...

Adding network interface card 'eth0' on cluster node 'lnsddb1' to the network 'db2_public_network_0' was successful.

Adding DB2 database partition '0' to the cluster ...

Adding DB2 database partition '0' to the cluster was successful.

HADR database 'TRADE6DB' has been determined to be valid for high availability.

However, the database cannot be added to the cluster from this node because

`db2haicu` detected this node is the standby for HADR database 'TRADE6DB'. Run

`db2haicu` on the primary for HADR database 'TRADE6DB' to configure the database for automated failover.

All cluster configurations have been completed successfully. `db2haicu` exiting ...

```
db2inst1@lnsddb3:~>
```

After the successful completion of **db2haicu** on both the primary and standby nodes, we now have a fully completed and integrated DB2 HADR cluster.

SA MP cluster verification

Upon successful completion of **db2haicu** on both nodes, SA MP will automatically define a cluster domain across the nodes and a virtual IP address on the primary node. SA MP monitors the DB2 instance and database behavior by running various DB2 scripts on a specified interval. The scripts are located under `/usr/sbin/rsct/sapolicies/db2` on both hosts in an HADR pair.

In Example 5-29, the `lssam` command is used to obtain a snapshot of the clustered resource states. A resource group can contain one or many resource group members, and each resource group contains one or many resources.

Example 5-29 Getting detailed information about SA MP resources

```

lnsudb1:/home/db2inst1 # lssam
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Nominal=Online
  |- Online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
    |- Online
      IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb1
        '- Offline
      IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb3
        '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs
          |- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb1
            '- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb3
Online IBM.ResourceGroup:db2_db2inst1_lnsudb1_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_lnsudb1_0-rs
    '- Online IBM.Application:db2_db2inst1_lnsudb1_0-rs:lnsudb1
Online IBM.ResourceGroup:db2_db2inst1_lnsudb3_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs
    '- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs:lnsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
  |- Online IBM.PeerNode:lnsudb1:lnsudb1
  '- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_db2inst1_lnsudb1_0-rg_group-equ
  '- Online IBM.PeerNode:lnsudb1:lnsudb1
Online IBM.Equivalency:db2_db2inst1_lnsudb3_0-rg_group-equ
  '- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:eth0:lnsudb3
  '- Online IBM.NetworkInterface:eth0:lnsudb1
lnsudb1:/home/db2inst1 #

```

We can see the cluster objects that are defined and running by SA MP in Example 5-29. The resource group for the HADR database pair, `trade6db`, is shown as online and the application definition for the virtual IP address, is confirmed using the `ifconfig` command as shown in Example 5-30.

Example 5-30 Verification of virtual IP on primary node

```

lnsudb1:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:0A:00:00:EC
          inet addr:9.12.7.86  Bcast:9.12.7.255  Mask:255.255.252.0
          inet6 addr: fe80::200:a00:900:ec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:478827 errors:0 dropped:0 overruns:0 frame:0
          TX packets:141112 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:43137620 (41.1 Mb)  TX bytes:38374576 (36.5 Mb)

eth0:0    Link encap:Ethernet  HWaddr 02:00:0A:00:00:EC
          inet addr:9.12.7.97  Bcast:9.12.7.255  Mask:255.255.252.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1

lo        Link encap:Local Loopback

```

```

inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:184 errors:0 dropped:0 overruns:0 frame:0
TX packets:184 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:24724 (24.1 Kb) TX bytes:24724 (24.1 Kb)

```

Another way to verify the state of the cluster is to check the DB2 memory sets, using the **db2pd** command. If the setup is successful, the command will return the domain details (such as quorum, VIP address, and so on), as shown in Example 5-31.

Example 5-31 Verification using db2pd

```

db2inst1@lnsddb1:~> db2pd -ha
          DB2 HA Status
Instance Information:
Instance Name           = db2inst1
Number Of Domains       = 1
Number Of RGs for instance = 2

Domain Information:
Domain Name             = itsodom1
Cluster Version         = 3.1.4.4
Cluster State           = Online
Number of nodes         = 2

Node Information:
Node Name               State
-----
lnsddb3                 Online
lnsddb1                 Online

Resource Group Information:
Resource Group Name     = db2_db2inst1_db2inst1_TRADE6DB-rg
Resource Group LockState = Unlocked
Resource Group OpState  = Online
Resource Group Nominal OpState = Online
Number of Group Resources = 2
Number of Allowed Nodes = 2
    Allowed Nodes
    -----
    lnsddb1
    lnsddb3

Member Resource Information:
Resource Name           = db2_db2inst1_db2inst1_TRADE6DB-rs
Resource State          = Online
Resource Type           = HADR
HADR Primary Instance   = db2inst1
HADR Secondary Instance = db2inst1
HADR DB Name            = TRADE6DB
HADR Primary Node       = lnsddb1
HADR Secondary Node     = lnsddb3

Resource Name           = db2ip_9_12_7_97-rs
Resource State          = Online

```

```

Resource Type                                = IP

Resource Group Name                          = db2_db2inst1_1nsudb1_0-rg
Resource Group LockState                     = Unlocked
Resource Group OpState                       = Online
Resource Group Nominal OpState               = Online
Number of Group Resources                     = 1
Number of Allowed Nodes                      = 1
  Allowed Nodes
  -----
  1nsudb1
Member Resource Information:
Resource Name                                = db2_db2inst1_1nsudb1_0-rs
Resource State                               = Online
Resource Type                                = DB2 Member
DB2 Member Number                           = 0
Number of Allowed Nodes                      = 1
  Allowed Nodes
  -----
  1nsudb1

Network Information:
Network Name                                Number of Adapters
-----
db2_public_network_0                      2

Node Name                                Adapter Name
-----
1nsudb3.itso.ibm.com                      eth0
1nsudb1.itso.ibm.com                      eth0

Quorum Information:
Quorum Name                                Quorum State
-----
db2_Quorum_Network_9_12_4_1:18_26_57      Online
db2_Quorum_Network_9_12_4_1:18_25_59      Offline
Success                                    Offline
Fail                                       Offline
Operator                                  Offline

```

5.2.5 SA MP DB2 automated failover scenarios

In this section, we simulate unplanned database failure scenarios and explain how SA MP manages the situation and provides a highly available cluster environment. These scenarios include unplanned failure scenarios of the DB2 instance on the HADR primary node, and an unplanned failure of the HADR primary cluster node itself.

Unplanned database instance failure

In DB2, there are commands that stop the instances such as **db2stop**. But those commands do a graceful shutdown of the database and our cluster manager (SA MP) would interpret the action as a planned manual outage and would not initiate an action on the failed node. So to simulate a database instance crash, we will use the **db2_kill** command to hard crash the database instance. In this case, the priority of cluster manager (SA MP) is to simply attempt to restart the DB2 instance that failed. In our case, it would be our primary node **1nsudb1**.

Example 5-32 displays the status of the standby database under normal working conditions.

Example 5-32 Before instance crash is simulated - normal cluster

```
db2inst1@lnsudb3:~> lssam
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
|- Pending online
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb1
'- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb3
'- Online IBM.ServiceIP:db2ip_9_12_7_97-rs
|- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb1
'- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb3
Online IBM.ResourceGroup:db2_db2inst1_lnsudb1_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst1_lnsudb1_0-rs
'- Online IBM.Application:db2_db2inst1_lnsudb1_0-rs:lnsudb1
Online IBM.ResourceGroup:db2_db2inst1_lnsudb3_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs
'- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs:lnsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
|- Online IBM.PeerNode:lnsudb1:lnsudb1
'- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_db2inst1_lnsudb1_0-rg_group-equ
'- Online IBM.PeerNode:lnsudb1:lnsudb1
Online IBM.Equivalency:db2_db2inst1_lnsudb3_0-rg_group-equ
'- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_public_network_0
|- Online IBM.NetworkInterface:eth0:lnsudb3
'- Online IBM.NetworkInterface:eth0:lnsudb1
```

Using the **db2_kill** command, as shown in Example 5-33, on the primary node results in a non-functioning primary node.

Example 5-33 Simulating Database Instance Crash using db2_kill command

```
db2inst1@lnsudb1:~> db2_kill
Application ipclean: Removing DB2 engine and client IPC resources for db2inst1.
```

At this point, the Tivoli System Automation (TSA) cluster manager senses that there is an inactive database instance state and notices that something has gone wrong with the primary instance. This is shown by using the **lssam** command output as shown in Example 5-34.

Example 5-34 lssam command verification

```
db2inst1@lnsudb3:~> lssam
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Nominal=Online
|- Pending online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
|- Pending online
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb1
'- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb3
'- Online IBM.ServiceIP:db2ip_9_12_7_97-rs
|- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb1
'- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb3
Pending online IBM.ResourceGroup:db2_db2inst1_lnsudb1_0-rg Nominal=Online
'- Pending online IBM.Application:db2_db2inst1_lnsudb1_0-rs
```

```

        '- Pending online
IBM.Application:db2_db2inst1_1nsudb1_0-rs:1nsudb1
Online IBM.ResourceGroup:db2_db2inst1_1nsudb3_0-rg Nominal=Online
        '- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs
            '- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs:1nsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
        |- Online IBM.PeerNode:1nsudb1:1nsudb1
        '- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_db2inst1_1nsudb1_0-rg_group-equ
        '- Online IBM.PeerNode:1nsudb1:1nsudb1
Online IBM.Equivalency:db2_db2inst1_1nsudb3_0-rg_group-equ
        '- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_public_network_0
        |- Online IBM.NetworkInterface:eth0:1nsudb3
        '- Online IBM.NetworkInterface:eth0:1nsudb1

```

During this time, database connectivity can not be established. This can be ascertained by repeatedly executing the database connect command and checking whether the connection can be established, as Example 5-35 demonstrates.

Example 5-35 connection attempts to database instance

Database Connection Information

```

Database server      = DB2/LINUXZ64 10.5.3
SQL authorization ID = DB2INST1
Local database alias = TRADE6DB

```

DB20000I The SQL command completed successfully.

```

SQL1032N No start database manager command was issued.  SQLSTATE=57019
SQL1024N A database connection does not exist.  SQLSTATE=08003

```

```

SQL1032N No start database manager command was issued.  SQLSTATE=57019
SQL1024N A database connection does not exist.  SQLSTATE=08003

```

During the various phases that TSA goes through during failover, given the current resource state, TSA would attempt to re-initiate a DB2 database instance recovery and reactivate the HADR primary database. The resource status is highlighted in Example 5-36.

Example 5-36 lssam reporting the change in the instance state

```

db2inst1@1nsudb3:~> lssam
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Request=Lock
Nominal=Online
        |- Pending online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
Control=StartInhibitedBecauseSuspended
        |- Pending online
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:1nsudb1
        '- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:1nsudb3
        '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs Control=SuspendedPropagated
            |- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb1
            '- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb3
Pending online IBM.ResourceGroup:db2_db2inst1_1nsudb1_0-rg Nominal=Online
        '- Pending online IBM.Application:db2_db2inst1_1nsudb1_0-rs

```

```

        '- Pending online
IBM.Application:db2_db2inst1_lnsudb1_0-rs:lnsudb1
Online IBM.ResourceGroup:db2_db2inst1_lnsudb3_0-rg Nominal=Online
        '- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs
            '- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs:lnsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
        |- Online IBM.PeerNode:lnsudb1:lnsudb1
        '- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_db2inst1_lnsudb1_0-rg_group-equ
        '- Online IBM.PeerNode:lnsudb1:lnsudb1
Online IBM.Equivalency:db2_db2inst1_lnsudb3_0-rg_group-equ
        '- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_public_network_0
        |- Online IBM.NetworkInterface:eth0:lnsudb3
        '- Online IBM.NetworkInterface:eth0:lnsudb1

```

In Example 5-37, we can see that TSA was successful in restarting the database instance and in reactivating the primary database into the HADR domain.

Example 5-37 Successful TSA Automation

```

db2inst1@lnsudb3:~> lssam
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Nominal=Online
    |- Pending online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
        |- Pending online
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb1
        '- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb3
        '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs
            |- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb1
            '- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb3
Online IBM.ResourceGroup:db2_db2inst1_lnsudb1_0-rg Nominal=Online
        '- Online IBM.Application:db2_db2inst1_lnsudb1_0-rs
            '- Online IBM.Application:db2_db2inst1_lnsudb1_0-rs:lnsudb1
Online IBM.ResourceGroup:db2_db2inst1_lnsudb3_0-rg Nominal=Online
        '- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs
            '- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs:lnsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
        |- Online IBM.PeerNode:lnsudb1:lnsudb1
        '- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_db2inst1_lnsudb1_0-rg_group-equ
        '- Online IBM.PeerNode:lnsudb1:lnsudb1
Online IBM.Equivalency:db2_db2inst1_lnsudb3_0-rg_group-equ
        '- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_public_network_0
        |- Online IBM.NetworkInterface:eth0:lnsudb3
        '- Online IBM.NetworkInterface:eth0:lnsudb1

db2inst1@lnsudb3:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
        |- Online
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb1
        '- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb3
        '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs

```



```

        |- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb1
        '- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb3
Online IBM.ResourceGroup:db2_db2inst1_1nsudb1_0-rg Nominal=Online
        '- Online IBM.Application:db2_db2inst1_1nsudb1_0-rs
        '- Online IBM.Application:db2_db2inst1_1nsudb1_0-rs:1nsudb1
Online IBM.ResourceGroup:db2_db2inst1_1nsudb3_0-rg Nominal=Online
        '- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs
        '- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs:1nsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
        |- Online IBM.PeerNode:1nsudb1:1nsudb1
        '- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_db2inst1_1nsudb1_0-rg_group-equ
        '- Online IBM.PeerNode:1nsudb1:1nsudb1
Online IBM.Equivalency:db2_db2inst1_1nsudb3_0-rg_group-equ
        '- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_public_network_0
        |- Online IBM.NetworkInterface:eth0:1nsudb3
        '- Online IBM.NetworkInterface:eth0:1nsudb1

```

Unplanned database node failure

In this scenario, we shut down the primary node to simulate a cluster node crash. If the primary database server fails abruptly, the network connectivity to that server will be severed from the cluster nodes as well as the external network.

DB2's Automatic Client Reroute (ACR) facility was first introduced in DB2 V8.2. This facility allows client applications to transparently connect to a standby server without exposing a communications error to the client applications.

Using the ACR, the DB2 client will ping the standby server and then the primary server, back and forth repeatedly, in an effort to see if one of them is alive and active. At this point, all the Tivoli System Automation manager has to do is detect a node failure (as shown in Example 5-38) and run the TAKEOVER HADR command. As we have configured Quorum device, it would act has to tie-breaker to avoid any split-brain scenarios.

Example 5-38 Primary node failure detection

```

db2inst1@1nsudb3:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Nominal=Online
        |- Online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
        |- Failed offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:1nsudb1
        '- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:1nsudb3
        '- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs
        |- Failed offline IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb1
        '- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb3
Online IBM.ResourceGroup:db2_db2inst1_1nsudb1_0-rg Nominal=Online
        '- Online IBM.Application:db2_db2inst1_1nsudb1_0-rs
        '- Failed offline
IBM.Application:db2_db2inst1_1nsudb1_0-rs:1nsudb1
Online IBM.ResourceGroup:db2_db2inst1_1nsudb3_0-rg Nominal=Online
        '- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs
        '- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs:1nsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
        |- Offline IBM.PeerNode:1nsudb1:1nsudb1

```

```

'- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_db2inst1_1nsudb1_0-rg_group-equ
'- Offline IBM.PeerNode:1nsudb1:1nsudb1
Online IBM.Equivalency:db2_db2inst1_1nsudb3_0-rg_group-equ
'- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_public_network_0
|- Online IBM.NetworkInterface:eth0:1nsudb3
'- Offline IBM.NetworkInterface:eth0:1nsudb1

db2inst1@1nsudb3:~> lssam
Pending online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg
Control=MemberInProblemState Nominal=Online
|- Offline IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
Control=MemberInProblemState
|- Failed offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:1nsudb1 Node=Offline
'- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:1nsudb3
'- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs Control=MemberInProblemState
|- Failed offline IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb1
Node=Offline
'- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb3
Failed offline IBM.ResourceGroup:db2_db2inst1_1nsudb1_0-rg Nominal=Online
'- Failed offline IBM.Application:db2_db2inst1_1nsudb1_0-rs
'- Failed offline
IBM.Application:db2_db2inst1_1nsudb1_0-rs:1nsudb1 Node=Offline
Online IBM.ResourceGroup:db2_db2inst1_1nsudb3_0-rg Nominal=Online
'- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs
'- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs:1nsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
|- Offline IBM.PeerNode:1nsudb1:1nsudb1 Node=Offline
'- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_db2inst1_1nsudb1_0-rg_group-equ
'- Offline IBM.PeerNode:1nsudb1:1nsudb1 Node=Offline
Online IBM.Equivalency:db2_db2inst1_1nsudb3_0-rg_group-equ
'- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_public_network_0
|- Online IBM.NetworkInterface:eth0:1nsudb3
'- Offline IBM.NetworkInterface:eth0:1nsudb1 Node=Offline

```

As in our case, TSA would detect the primary node has failed, it would run the **TAKEOVER HADR ON DATABASE prod BY FORCE** command which will cause the standby to become a primary server. The standby will apply the last log buffers that it has, undo any in-flight transactions, and then open the database for connections. At this point the automatic client reroute will successfully establish a connection from the client application to the standby (now primary) server and inform the application that the most recent in-flight transaction has been rolled back. The transition is represented in TSAM as in Example 5-39.

Example 5-39 TSA Transitioning standby database as primary

```

db2inst1@1nsudb3:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Request=Lock
Nominal=Online
|- Online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
Control=SuspendedPropagated

```

```

|- Failed offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:1nsudb1 Node=Offline
  '- Online
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:1nsudb3
  '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs Control=SuspendedPropagated
    |- Failed offline IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb1
Node=Offline
  '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:1nsudb3
Failed offline IBM.ResourceGroup:db2_db2inst1_1nsudb1_0-rg Nominal=Online
  '- Failed offline IBM.Application:db2_db2inst1_1nsudb1_0-rs
    '- Failed offline
IBM.Application:db2_db2inst1_1nsudb1_0-rs:1nsudb1 Node=Offline
Online IBM.ResourceGroup:db2_db2inst1_1nsudb3_0-rg Nominal=Online
  '- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs
    '- Online IBM.Application:db2_db2inst1_1nsudb3_0-rs:1nsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
  |- Offline IBM.PeerNode:1nsudb1:1nsudb1 Node=Offline
  '- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_db2inst1_1nsudb1_0-rg_group-equ
  '- Offline IBM.PeerNode:1nsudb1:1nsudb1 Node=Offline
Online IBM.Equivalency:db2_db2inst1_1nsudb3_0-rg_group-equ
  '- Online IBM.PeerNode:1nsudb3:1nsudb3
Online IBM.Equivalency:db2_public_network_0
  |- Online IBM.NetworkInterface:eth0:1nsudb3
  '- Offline IBM.NetworkInterface:eth0:1nsudb1 Node=Offline

```

Apart from changing the state of the standby database to primary, it also defines the Virtual IP address that was defined in the primary server, confirmed in Example 5-40. This would ensure that applications would not lose connection to the database even in the case of a primary node failure.

Example 5-40 Virtual IP floated to Standby server by TSA

```

1nsudb3:~ # ifconfig
eth0      Link encap:Ethernet  HWaddr 02:00:0A:00:00:F6
          inet addr:9.12.7.90  Bcast:9.12.7.255  Mask:255.255.252.0
          inet6 addr: fe80::200:a00:1000:f6/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:24160 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5789 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2760237 (2.6 Mb)  TX bytes:964429 (941.8 Kb)

eth0:0    Link encap:Ethernet  HWaddr 02:00:0A:00:00:F6
          inet addr:9.12.7.97  Bcast:9.12.7.255  Mask:255.255.252.0
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:114 errors:0 dropped:0 overruns:0 frame:0
          TX packets:114 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:9316 (9.0 Kb)  TX bytes:9316 (9.0 Kb)

```

In Example 5-41, we can see that the standby node has taken up the role of primary and started to accept database requests. This transition may take some time (in seconds) for the status to get reflected due to the network latency in applying the log buffers, during which the standby server will show **State = Remote catchup pending**.

Example 5-41 snapshot of TSA states

```
db2inst1@lnsudb1:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
    |- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb1
    '- Online
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsudb3
    '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs
        |- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb1
        '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsudb3
Online IBM.ResourceGroup:db2_db2inst1_lnsudb1_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_lnsudb1_0-rs
        '- Online IBM.Application:db2_db2inst1_lnsudb1_0-rs:lnsudb1
Online IBM.ResourceGroup:db2_db2inst1_lnsudb3_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs
        '- Online IBM.Application:db2_db2inst1_lnsudb3_0-rs:lnsudb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
    |- Online IBM.PeerNode:lnsudb1:lnsudb1
    '- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_db2inst1_lnsudb1_0-rg_group-equ
    '- Online IBM.PeerNode:lnsudb1:lnsudb1
Online IBM.Equivalency:db2_db2inst1_lnsudb3_0-rg_group-equ
    '- Online IBM.PeerNode:lnsudb3:lnsudb3
Online IBM.Equivalency:db2_public_network_0
    |- Online IBM.NetworkInterface:eth0:lnsudb3
    '- Online IBM.NetworkInterface:eth0:lnsudb1
```

As soon as the primary node comes back online, it will contact the standby node to say it is alive and the primary will automatically resynchronize itself. After it is back in a peer state, the standby node will again ship log buffers to the primary node. So a failure of the primary node or network does not cause a failure of the standby server.

In Example 5-42 we can see that when the primary DB2 node comes up, the current primary node initiates the re-synchronization of logs between the nodes. When the logs are synchronized, the **HADR_STATE** (the high availability disaster recovery state of the database) will change to **PEER**.

Example 5-42 primary node log synchronization

```
db2inst1@lnsudb3:~> db2pd -hadr -db trade6db
DatabaseMember0--DatabaseTRADE6DB--Active--Up0days00:00:24--Date2
014-06-24-16.44.54.755929

        HADR_ROLE = PRIMARY
        REPLAY_TYPE = PHYSICAL
        HADR_SYNCMODE = NEARSYNC
        STANDBY_ID = 1
        LOG_STREAM_ID = 0
        HADR_STATE = PEER
```

```

HADR_FLAGS = STANDBY_LOG_RETRIEVAL
PRIMARY_MEMBER_HOST = lnsudb3
PRIMARY_INSTANCE = db2inst1
PRIMARY_MEMBER = 0
STANDBY_MEMBER_HOST = lnsudb1
STANDBY_INSTANCE = db2inst1
STANDBY_MEMBER = 0
HADR_CONNECT_STATUS = CONNECTED
HADR_CONNECT_STATUS_TIME = 06/24/2014 16:44:33.418624 (1403642673)
HEARTBEAT_INTERVAL(seconds) = 30
HADR_TIMEOUT(seconds) = 120
TIME_SINCE_LAST_RECV(seconds) = 21
PEER_WAIT_LIMIT(seconds) = 0
LOG_HADR_WAIT_CUR(seconds) = 0.000
LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
LOG_HADR_WAIT_COUNT = 0
SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
PRIMARY_LOG_FILE,PAGE,POS = S0000021.LOG, 0, 182670062
STANDBY_LOG_FILE,PAGE,POS = S0000021.LOG, 0, 182670062
HADR_LOG_GAP(bytes) = 0
STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000021.LOG, 0, 182670062
STANDBY_RECV_REPLAY_GAP(bytes) = 0
PRIMARY_LOG_TIME = 06/24/2014 09:17:09.000000 (1403615829)
STANDBY_LOG_TIME = 06/24/2014 09:17:09.000000 (1403615829)
STANDBY_REPLAY_LOG_TIME = 06/24/2014 09:17:09.000000 (1403615829)
STANDBY_RECV_BUF_SIZE(pages) = 982
STANDBY_RECV_BUF_PERCENT = 0
STANDBY_SPOOL_LIMIT(pages) = 0
STANDBY_SPOOL_PERCENT = NULL
PEER_WINDOW(seconds) = 300
PEER_WINDOW_END = 06/24/2014 16:49:33.000000 (1403642973)
READS_ON_STANDBY_ENABLED = N

```

In Example 5-43, we can see that logs have been synchronized and the TSA cluster manager initiates a fall back from standby to primary node. The HADR state of the standby node is changed to STANDBY. Apart from the node takeover, TSA removes the virtual IP from standby node and creates the virtual IP in the primary node.

Example 5-43 automatic fallback from standby to primary

```
db2inst1@lnsudb1:~> db2pd -hadr -db trade6db
```

```
Database Member 0 -- Database TRADE6DB -- Active -- Up 0 days 00:18:00 -- Date
2014-06-24-17.01.55.292928
```

```

HADR_ROLE = PRIMARY
REPLAY_TYPE = PHYSICAL
HADR_SYNCMODE = NEARSYNC
STANDBY_ID = 1
LOG_STREAM_ID = 0
HADR_STATE = PEER
HADR_FLAGS =
PRIMARY_MEMBER_HOST = lnsudb1
PRIMARY_INSTANCE = db2inst1

```

```

        PRIMARY_MEMBER = 0
        STANDBY_MEMBER_HOST = lnsudb3
        STANDBY_INSTANCE = db2inst1
        STANDBY_MEMBER = 0
        HADR_CONNECT_STATUS = CONNECTED
        HADR_CONNECT_STATUS_TIME = 06/24/2014 16:44:33.418826 (1403642673)
        HEARTBEAT_INTERVAL(seconds) = 30
        HADR_TIMEOUT(seconds) = 120
        TIME_SINCE_LAST_RECV(seconds) = 5
        PEER_WAIT_LIMIT(seconds) = 0
        LOG_HADR_WAIT_CUR(seconds) = 0.000
        LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
        LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
        LOG_HADR_WAIT_COUNT = 0
        SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
        SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
        PRIMARY_LOG_FILE,PAGE,POS = S0000021.LOG, 11, 182715042
        STANDBY_LOG_FILE,PAGE,POS = S0000021.LOG, 11, 182715042
        HADR_LOG_GAP(bytes) = 0
        STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000021.LOG, 11, 182715042
        STANDBY_RECV_REPLAY_GAP(bytes) = 0
        PRIMARY_LOG_TIME = 06/24/2014 16:59:33.000000 (1403643573)
        STANDBY_LOG_TIME = 06/24/2014 16:59:33.000000 (1403643573)
        STANDBY_REPLAY_LOG_TIME = 06/24/2014 16:59:33.000000 (1403643573)
        STANDBY_RECV_BUF_SIZE(pages) = 16
        STANDBY_RECV_BUF_PERCENT = 0
        STANDBY_SPOOL_LIMIT(pages) = 25600
        STANDBY_SPOOL_PERCENT = 0
        PEER_WINDOW(seconds) = 300
        PEER_WINDOW_END = 06/24/2014 17:06:50.000000 (1403644010)
        READS_ON_STANDBY_ENABLED = Y
        STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N

```

db2inst1@lnsudb3:~> db2pd -hadr -db trade6db

Database Member 0 -- Database TRADE6DB -- Active Standby -- Up 0 days 00:17:32 --
Date 2014-06-24-17.02.02.154259

```

        HADR_ROLE = STANDBY
        REPLAY_TYPE = PHYSICAL
        HADR_SYNCMODE = NEARSYNC
        STANDBY_ID = 0
        LOG_STREAM_ID = 0
        HADR_STATE = PEER
        HADR_FLAGS =
        PRIMARY_MEMBER_HOST = lnsudb1
        PRIMARY_INSTANCE = db2inst1
        PRIMARY_MEMBER = 0
        STANDBY_MEMBER_HOST = lnsudb3
        STANDBY_INSTANCE = db2inst1
        STANDBY_MEMBER = 0
        HADR_CONNECT_STATUS = CONNECTED
        HADR_CONNECT_STATUS_TIME = 06/24/2014 16:44:33.418624 (1403642673)
        HEARTBEAT_INTERVAL(seconds) = 30
        HADR_TIMEOUT(seconds) = 120
        TIME_SINCE_LAST_RECV(seconds) = 11

```

```

        PEER_WAIT_LIMIT(seconds) = 0
        LOG_HADR_WAIT_CUR(seconds) = 0.000
        LOG_HADR_WAIT_RECENT_AVG(seconds) = 0.000000
        LOG_HADR_WAIT_ACCUMULATED(seconds) = 0.000
        LOG_HADR_WAIT_COUNT = 0
    SOCK_SEND_BUF_REQUESTED,ACTUAL(bytes) = 0, 16384
    SOCK_RECV_BUF_REQUESTED,ACTUAL(bytes) = 0, 87380
        PRIMARY_LOG_FILE,PAGE,POS = S0000021.LOG, 11, 182715042
        STANDBY_LOG_FILE,PAGE,POS = S0000021.LOG, 11, 182715042
        HADR_LOG_GAP(bytes) = 0
    STANDBY_REPLAY_LOG_FILE,PAGE,POS = S0000021.LOG, 11, 182715042
    STANDBY_RECV_REPLAY_GAP(bytes) = 0
        PRIMARY_LOG_TIME = 06/24/2014 16:59:33.000000 (1403643573)
        STANDBY_LOG_TIME = 06/24/2014 16:59:33.000000 (1403643573)
        STANDBY_REPLAY_LOG_TIME = 06/24/2014 16:59:33.000000 (1403643573)
    STANDBY_RECV_BUF_SIZE(pages) = 982
    STANDBY_RECV_BUF_PERCENT = 0
    STANDBY_SPOOL_LIMIT(pages) = 25600
    STANDBY_SPOOL_PERCENT = 0
    PEER_WINDOW(seconds) = 300
        PEER_WINDOW_END = 06/24/2014 17:06:50.000000 (1403644010)
    READS_ON_STANDBY_ENABLED = Y
    STANDBY_REPLAY_ONLY_WINDOW_ACTIVE = N

```

In the foregoing scenarios, the total time taken for TSA to run the takeover is seconds. In addition to that, there is little to no downtime period after the failure and many applications likely will not notice a problem.



Summary

This chapter provides a summary of the hardware and software solutions for making applications highly available when the host platform is System z. We review the high availability options for workload running on Linux on System z, z/OS, or end-to-end applications that include both operating systems.

This chapter covers the following topics:

- ▶ The need for high availability
- ▶ Applications that need to be highly available
- ▶ High availability options available on System z

6.1 The need for high availability

In Chapter 2, “Business and IT availability” on page 11, we discuss the business issues that are involved in a decision about availability requirements. Can you afford not to have HA? No company needs to have every system and every application highly available. But, most companies have some applications that are critical to their success and need to have almost continuous availability.

Business requirements usually drive the need for implementing high availability environments. Another reason can be legal requirements and regulations. Government regulations can specify how organizations must handle data and business processes. Also, application and system complexity can create a need for redundant systems. As application paths cross networks, platforms and multiple layers of software, the number of components increases, increasing the likelihood of breakage somewhere in the application path.

A solution must be identified that balances the costs of the solution with the financial impact of an outage. Your company needs to calculate the impact of an outage in your specific case. Also, for complex application systems with many components, it is important to identify the pieces that are mostly likely to cause an outage and implement redundancy for them.

6.2 Applications that need to be highly available

If your organization has service level agreements (SLA) with response time or availability requirements, that is a good place to start when deciding which applications need to be highly available. Additionally, SLAs can provide input as to the level of availability needed, and how much outage (or none) can be tolerated.

Most businesses have some level of backup or redundancy for key applications. It is important to re-evaluate those systems periodically or when major changes are made to ensure that new single points of failure have not been introduced.

It is important to keep in mind that there is no single best solution. All methods have their advantages and drawbacks that need to be considered and a balance found for the most suitable solution, taking into consideration all technical and non-technical aspects. This applies to the disparate systems inside an IT organization as well as across companies.

6.3 High availability options available on System z

In 3.1, “Components of a high availability solution” on page 30, we discuss the many hardware, software, and networking options available for applications running on System z. The number of requirements and complexity involved in creating highly available systems expand for today’s end-to-end applications. Fortunately, System z offers a broad range of solutions for Linux and z/OS.

Linux operating systems include a number of features for improving availability. On System z, this also affords the option of virtualizing Linux systems under z/VM. Running under z/VM, the Linux guests can take advantage of the hypervisor clustering feature, Single System Image, and the high availability feature for moving Linux images (called Live Guest Relocation) from one system to another. z/VM’s robust HA features can greatly reduce planned and unplanned outages for Linux systems.

End-to-end applications frequently cross platforms or operating systems such as mainframe z/OS and Linux. When Linux and z/OS cohabitate on System z, you can take advantage of the specialized high availability hardware and software of that environment.

Planning for high availability often includes setting up multiple data centers to avoid the possibility of an outage caused by a natural disaster. Applications running on System z can be spread across two or more sites through the functionality offered by IBM's Geographically Dispersed Parallel Sysplex solution (GDPS). GDPS® is a multi-site or single-site end-to-end application availability solution that provides the capability to manage remote copy configuration and storage subsystems (including IBM TotalStorage Enterprise Storage Server), to automate operational tasks and perform failure recovery from a single point of control.

The decision of whether or not to create redundant systems for high availability is generally an easy one for most businesses. Deciding which applications require HA solutions, what components need redundancy, and how to implement HA, is more difficult. This IBM Redbooks publication discusses many of the business issues and HA options that you will need to consider as part of the process to improve your IT resilience.



A

Common TSA commands

This appendix discusses the most common Tivoli System Automation (TSA) commands used for clustering.

Common Tivoli system automation commands

Some commonly used system automation commands are presented in Table A-1.

Table A-1 TSA commands

Command	Description
preprnode	This command prepares the security settings for the node to be included in a cluster. When issued, public keys are exchanged among the nodes, and the RMC access control list (ACL) is modified to enable access to cluster resources by all the nodes of the cluster.
mkrpdomain	This command creates a new cluster definition. It is used to specify the name of the cluster, and the list of nodes to be added to the cluster.
startprdomain / stopprdomain	These commands are used to bring the cluster online and offline, respectively.
addrpnode	After a cluster has been defined and is operational, this command is used to add new nodes to the cluster.
startprnode / stopprnode	These commands are used to bring individual nodes online and offline to the cluster. They are often used when performing maintenance to a particular system. The node is stopped, repairs or maintenance is performed, then the node is restarted, at which time it rejoins the cluster.
lsrpnnode	This command is used to view the list of nodes defined to a cluster, as well as the operating state (OpState) of each node. Note that this command is useful only on nodes that are Online in the cluster; otherwise it will not display the list of nodes.
rmrpdomain	This command removes a cluster definition from the nodes.
rmrpnnode	This command removes one or more nodes from a cluster definition.



B

Hints and tips during cluster setup

This appendix discusses some of the common hints and tips we learned during the TSA-based DB2 HADR Cluster setup.

Domain nodes not communicating

When running the **db2haicu** command to create and define a domain and each node, the **db2haicu** command would fail during the inclusion of nodes into the domain. To correct this, do the following checks:

1. Check whether RSCT on both nodes is started and both are the same version level.
2. Check whether the **preprnode** command has been executed on both nodes.
3. Verify the version of TSA cluster scripts on both nodes. They must be the same version. If not, this may lead to unpredictable results.
4. Verify whether the license has been applied for TSA. Sometimes this can also create issues while creating domains.

Node names

Make sure that the servers' (primary and standby) hostnames are consistently defined in the following places:

- ▶ The Linux `/etc/hosts` file
- ▶ The DB2 HADR configuration parameters (`db2pd -hadr -db trade6db`)
- ▶ The DB2 node configuration file - `db2nodes.cfg` (in `$HOME/sqllib`)
- ▶ During **db2haicu** cluster node definitions

Problems with db2haicu definition

If you need to redefine the current **db2haicu** cluster domain definitions or if you need to add other nodes to the cluster, use the **db2haicu --delete** command to fully remove the domain definitions from the nodes. The command must be executed on both nodes so that the configuration files are removed properly. After the domain nodes are fully removed, use the **lssam** command to verify if it has removed the domain definition.

Unhealthy TSA cluster state

During the cluster configuration phase, we mistakenly executed a few commands that caused TSA to try to do failover, even when the nodes were up and running. Most of the time TSA would be able to cope with this kind of situation, but some failures can cause TSA to get stuck in a "pending online" or other unhealthy state.

In this situation, you can stop the TSA cluster configuration and remove the cluster nodes. Then the database administrator can perform the tasks to solve any database synchronization issue. After the issue with the DB2 database synchronization is solved, you can continue to start the HADR from DB2 and TSA clustering.

To do this, perform the following steps:

1. Disable DB2 integration with SA MP on each DB2 server. Execute the command in both the primary and standby nodes, as shown in Example B-1.

Example B-1 Disabling high-availability for DB2

```
db2inst1@lnsddb1:~> db2haicu -disable
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).
```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is 'db2inst1'. The cluster configuration that follows will apply to this instance.

Are you sure you want to disable high availability (HA) for the database instance 'db2inst1'. This will lock all the resource groups for the instance and disable the HA configuration parameter. The instance will not failover if a system outage occurs while the instance is disabled. You will need to run db2haicu again to enable the instance for HA. Disable HA for the instance 'db2inst1'? [1]

1. Yes

2. No

1

```
Disabling high availability for instance 'db2inst1' ...
Locking the resource group for HADR database 'TRADE6DB' ...
Locking the resource group for HADR database 'TRADE6DB' was successful.
Locking the resource group for DB2 database partition '0' ...
Locking the resource group for DB2 database partition '0' was successful.
Locking the resource group for DB2 database partition '0' ...
Locking the resource group for DB2 database partition '0' was successful.
Disabling high availability for instance 'db2inst1' was successful.
All cluster configurations have been completed successfully. db2haicu exiting ...
```

2. Force the domain offline and ensure that it moves from Online to Offline using the **lsrpdomain** command, as demonstrated in Example B-2.

Example B-2 Stopping the cluster domain and verifying the domain

```
lnsddb1:~ # stoprpdomain -f itsodom1
lnsddb1:~ # lsrpdomain
Name      OpState RSCTActiveVersion MixedVersions TSPort GSPort
itsodom1  Offline 3.1.4.4          No           12347 12348
```

```
db2inst1@lnsddb1:~> lssam
lssam: No resource groups defined or cluster is offline!
```

```
db2inst1@lnsddb1:~> lssam
lssam: No resource groups defined or cluster is offline!
```

At this point, the TSA cluster is totally decoupled from the DB2 nodes. Any measures the DB2 administrator wants to take on the database can be done, and after the database activities are completed, we can proceed with re-integrating the nodes with the TSA cluster as in Example B-3.

Example B-3 Starting the cluster domain

```

lnsddb1:~ # startrpdomain itsodom1

lnsddb1:~ # lsrpdomain
Name      OpState RSCTActiveVersion MixedVersions TSPort GSPort
itsodom1 Online 3.1.4.4          No           12347 12348

```

After a few seconds, we can check whether the defined nodes have been included and online with the **lsrpnode** command. The results are shown in Example B-4.

Example B-4 checking on the domain nodes

```

lnsddb1:~ # lsrpnode
Name      OpState RSCTVersion
lnsddb3 Online 3.1.4.4
lnsddb1 Online 3.1.4.4

```

3. Enable HA integration for each DB2 instance as demonstrated in Example B-5.

Example B-5 Execute db2haicu command on both the nodes

```

db2inst1@lnsddb1:~> db2haicu
Welcome to the DB2 High Availability Instance Configuration Utility (db2haicu).

```

You can find detailed diagnostic information in the DB2 server diagnostic log file called db2diag.log. Also, you can use the utility called db2pd to query the status of the cluster domains you create.

For more information about configuring your clustered environment using db2haicu, see the topic called 'DB2 High Availability Instance Configuration Utility (db2haicu)' in the DB2 Information Center.

db2haicu determined the current DB2 database manager instance is 'db2inst1'. The cluster configuration that follows will apply to this instance.

db2haicu is collecting information on your current setup. This step may take some time as db2haicu will need to activate all databases for the instance to discover all paths ...

When you used db2haicu to configure your clustered environment, you create cluster domains. For more information, see the topic 'Creating a cluster domain with db2haicu' in the DB2 Information Center. db2haicu is searching the current machine for an existing active cluster domain ...

db2haicu found a cluster domain called 'itsodom1' on this machine. The cluster configuration that follows will apply to this domain.

db2haicu has detected that high availability has been disabled for the instance 'db2inst1'. Do you want to enable high availability for the instance 'db2inst1'?

```

[1]
1. Yes
2. No
1

```

```

Retrieving high availability configuration parameter for instance 'db2inst1' ...
The cluster manager name configuration parameter (high availability configuration
parameter) is not set. For more information, see the topic "cluster_mgr - Cluster
manager name configuration parameter" in the DB2 Information Center. Do you want
to set the high availability configuration parameter?
The following are valid settings for the high availability configuration
parameter:
    1.TSA
    2.Vendor
Enter a value for the high availability configuration parameter: [1]
1
Setting a high availability configuration parameter for instance 'db2inst1' to
'TSA'.
Enabling high availability for instance 'db2inst1' ...
Enabling high availability for instance 'db2inst1' was successful.
All cluster configurations have been completed successfully. db2haicu exiting ...
db2inst1@lnsldb1:~>

```

4. Now use the **lssam** command to ensure that the cluster state has changed to online and the Virtual IP is defined on the primary node. The expected state is shown in Example B-6.

Example B-6 verifying that the cluster is back online

```

db2inst1@lnsldb3:~> lssam
Online IBM.ResourceGroup:db2_db2inst1_db2inst1_TRADE6DB-rg Nominal=Online
    |- Online IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs
        |- Online
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsldb1
    '- Offline
IBM.Application:db2_db2inst1_db2inst1_TRADE6DB-rs:lnsldb3
    '- Online IBM.ServiceIP:db2ip_9_12_7_97-rs
        |- Online IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsldb1
        '- Offline IBM.ServiceIP:db2ip_9_12_7_97-rs:lnsldb3
Online IBM.ResourceGroup:db2_db2inst1_lnsldb1_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_lnsldb1_0-rs
        '- Online IBM.Application:db2_db2inst1_lnsldb1_0-rs:lnsldb1
Online IBM.ResourceGroup:db2_db2inst1_lnsldb3_0-rg Nominal=Online
    '- Online IBM.Application:db2_db2inst1_lnsldb3_0-rs
        '- Online IBM.Application:db2_db2inst1_lnsldb3_0-rs:lnsldb3
Online IBM.Equivalency:db2_db2inst1_db2inst1_TRADE6DB-rg_group-equ
    |- Online IBM.PeerNode:lnsldb1:lnsldb1
    '- Online IBM.PeerNode:lnsldb3:lnsldb3
Online IBM.Equivalency:db2_db2inst1_lnsldb1_0-rg_group-equ
    '- Online IBM.PeerNode:lnsldb1:lnsldb1
Online IBM.Equivalency:db2_db2inst1_lnsldb3_0-rg_group-equ
    '- Online IBM.PeerNode:lnsldb3:lnsldb3
Online IBM.Equivalency:db2_public_network_0
    |- Online IBM.NetworkInterface:eth0:lnsldb3
    '- Online IBM.NetworkInterface:eth0:lnsldb1
db2inst1@lnsldb3:~>

```

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks publications

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only:

- ▶ *IBM System z Connectivity Handbook*, SG24-5444
- ▶ *Achieving High Availability on Linux for System z with Linux-HA Release 2*, SG24-7711
- ▶ *End to End Automation with IBM Tivoli System Automation for Multiplatforms*, SG24-7117
- ▶ *GDPS Family an Introduction to Concepts and Facilities*, SG24-6374

You can search for, view, download or order these documents and other Redbooks publications, Redpaper publications, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ *z/VM: TCP/IP Planning and Customization*, SC24-6238-04
- ▶ *z/VM: Connectivity book*, SC24-6174-04
- ▶ *z/VM: Commands and Utilities Reference*, SC24-6166

Help from IBM

IBM Support and downloads:

ibm.com/support

IBM Global Services:

ibm.com/services



End-to-End High Availability Solution for System z from a Linux Perspective

Achieves proactive high availability of heterogeneous environments

Covers multiplatforms: Linux on System z, z/VM, and z/OS

Includes architectural scenarios and practical applications

As Linux on System z becomes more prevalent and mainstream in the industry, the need for it to deliver higher levels of availability is increasing.

This IBM Redbooks publication starts with an explanation of high availability (HA) fundamentals such as HA concepts and terminology. It continues with a discussion of why a business needs to consider an HA solution and then explains how to determine your business single points of failure.

We outline the components of a high availability solution and describe these components. Then we provide some architectural scenarios and demonstrate how to plan and decide an implementation of an end-to-end HA solution, from Linux on System z database scenarios to z/OS, and include storage, network, z/VM, Linux, and middleware.

This implementation includes the IBM Tivoli System Automation for Multiplatforms (TSA MP), which monitors and automates applications distributed across Linux, AIX, and z/OS operating systems, as well as a GDPS based solution. It includes the planning for an end-to-end scenario, considering Linux on System z, z/VM, and z/OS operating environments, and the middleware used.

The TSA MP implements HA for infrastructure, network, operating systems, and applications across multiple platforms and is compared to a Linux HA implementation based on open source Linux-HA, which is Linux only.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks