ibm.com/redbooks

# Using Pacemaker to Create Highly Available Linux Solutions on IBM Power

Tim Simon

Dishant Doriwala

Hemantha Gunasinghe

Munshi Hafizul Haque

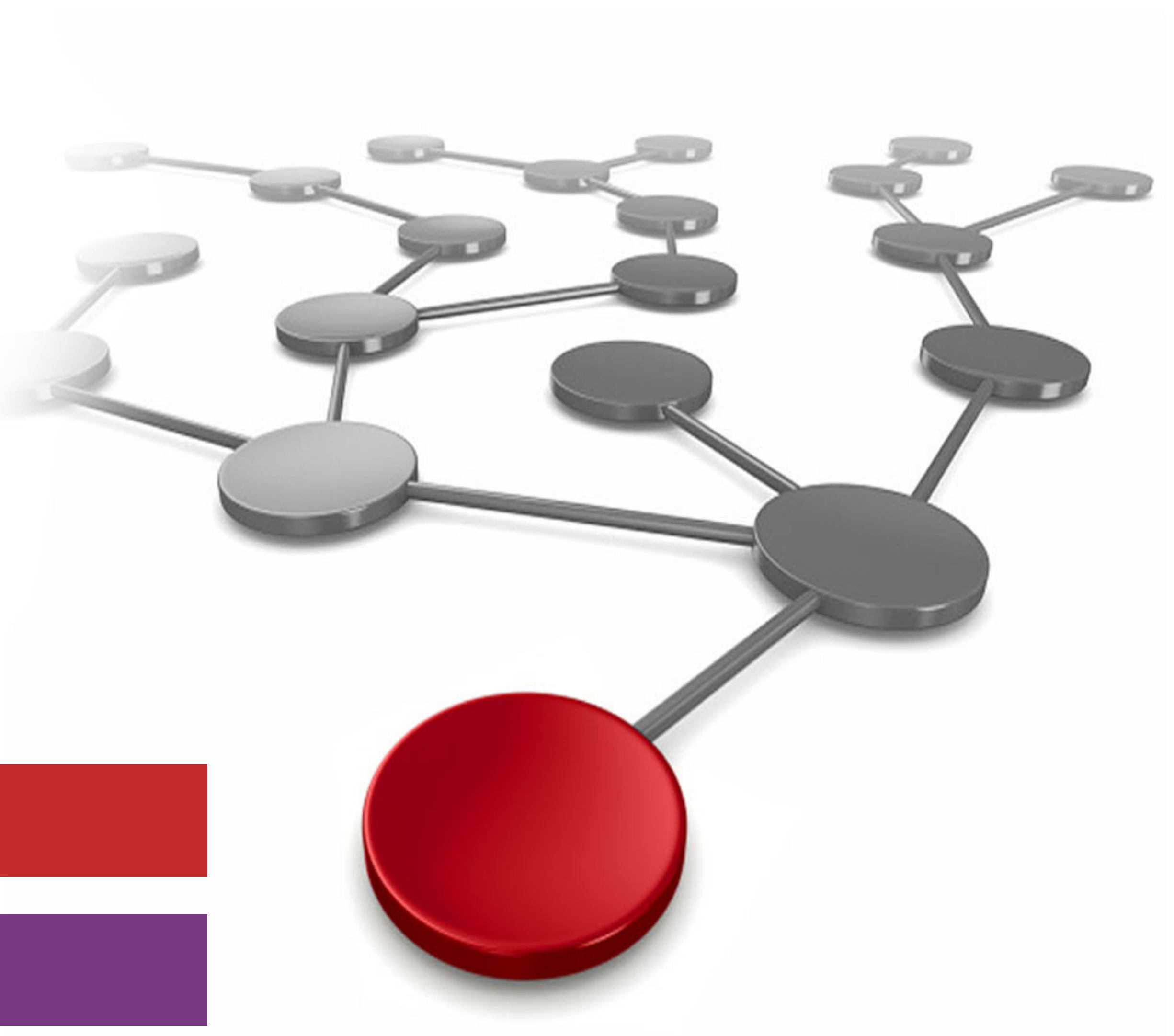
Santhosh Joshi

Adhish Kapoor

Pravin Kedia

Youssef Largou

Henry Vo



Cloud

Linux

IBM Redbooks

# Using Pacemaker to Create Highly Available Linux Solutions on IBM Power

October 2024

SG24-8557-00

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

**First Edition (October 2024)**

This edition applies to the following software and hardware versions:
Red Hat Enterprise Linux 9.0 for IBM Power Little Endian
Red Hat Enterprise Linux 8.2 for IBM Power Little Endian
SUSE Linux Enterprise Server 15 SP1
DB2 Version 11.5 FP8
IBM MQ 9.2 LTS
IBM MQ 9.2 LT3
IBM VM Recovery Manager Version 1.7
IBM PowerVC 2.1.0
IBM Power 10
IBM Power 9

# Contents

# Notices

## Trademarks

# Preface

High availability and business continuity continue to be focus themes for businesses as they work hard to provide infrastructure components to support the "always available" requirements for their critical business functions. In addition, many enterprises are starting to build critical business functions on Linux servers to take advantage of the openness and portability of those applications across both on-premises and cloud-based infrastructures. These combined requirements lead to the need for a solution that allows enterprises to support highly available Linux environments for critical business applications.

This IBM Redbooks publication will help you install, tailor and configure a highly available cluster of IBM Power servers utilizing Pacemaker, which is an open source cluster manager that supports Linux environments. We describe basic concepts required for planning and designing an infrastructure to meet the requirements for high availability and business continuity and we compare and contrast different options and tools. While the publication is mostly targeted at skilled professionals who are designing and installing highly available Linux environments, it starts with a basic understanding of the technology to help less technical readers understand the concepts.

## Authors

This book was produced by a team of specialists from around the world working at IBM Redbooks, Poughkeepsie Center.

**Tim Simon** is an IBM® Redbooks® Project Leader in Tulsa, Oklahoma, USA. He has over 40 years of experience with IBM, primarily in a technical sales role working with customers to help them create IBM solutions to solve their business problems. He holds a BS degree in Math from Towson University in Maryland. He has worked with many IBM products and has extensive experience creating customer solutions using IBM Power, IBM Storage, and IBM System z® throughout his career. He currently lives in Tulsa, Oklahoma where he enjoys spending time with his grandchildren.

**Dishant Doriwala** is a Senior Staff Software engineer, Test Lead for VM Recovery Manager (HA and DR solution for Power Systems). He works in IBM Systems Development Labs (ISDL), Hyderabad, India and has 11 years of experience in the industry with expertise in testing for HA-DR products. He has experience working on various HA and DR solutions: IBM PowerHA System Mirror, Reliable Scalable Cluster Technology, VM Recovery Manager and GLVM. He has expertise on enterprise storage: IBM San Volume controller/Storwize®, Hitachi, Dell EMC SRDF and Unity XT, IBM XIV®. He has authored technical white papers, Redpapers, and conference articles in the HA-DR domains. He holds a masters degree in Software Technologies from International Institute of Information Technology, Pune and a bachelors degree in Electronics and Communications from VTU, Bangalore.

**Hemantha Gunasinghe** is an IT Specialist in IBM Systems Expert Labs within IBM Australia. He brings a unique experience and perspective with over 32 years of experience in the IT industry in Sri Lanka and Australia. Before migrating to Australia in 1999, Hemantha was working as a Systems Engineer at IBM Sri Lanka. Hemantha joined IBM Australia as a Service Delivery administrator in IBM Global Services for IBM POWER® systems where he provided many managed infrastructure services to IBM Internal systems, as well as for commercial customers at various capacities in the area of IBM AIX®, IBM PowerHA®, GPFS, Linux, and IBM PowerVM® Virtualization technologies on IBM POWER systems.

He joined IBM LAB Services in 2013 as an IT Management Consultant and has delivered many IT projects and services to IBM clients on IBM POWER systems. He has also provided implementation and migration services to IBM Power VS in IBM Cloud®. Recently, he has delivered some of the largest SAP HANA systems running on Linux on Power using Red Hat Operating systems and has also delivered OpenShift clusters deployed on Power Systems. His services extend beyond Australia into countries like New Zealand and the United States, where major government, defense, commercial, and financial institutions collaborate with him on implementation and building services for their clients. He holds a degree in Mathematics, Statistics, and Computing from The University of Kelaniya in Sri Lanka with 1st class honors, as well as a masters degree in computer science from The University of Colombo in Sri Lanka.

**Munshi Hafizul Haque** is a Senior Platform Consultant at Red Hat in Kuala Lumpur, Malaysia. Munshi is an experienced technologist in engineering, design, and architecture of PaaS and cloud infrastructures. He has over 16 years of experience in post-delivery technical support, designing full infrastructure solutions, performing installation, implementation and integration services of systems, storage and software solutions, and conducting post implementation assessments and presentations at customer engagements and corporate events. He is currently part of the Red Hat Consulting Services team where he helps organizations adopt automation, container technology, and DevOps practices. Before that, he worked for IBM as a senior consultant with IBM Systems Lab Services in Petaling Jaya, Malaysia, where he took part in various projects with different people in different ASEAN countries, and as a specialist in IBM Power Systems and associated enterprise edition technology.

**Santhosh Joshi** is a Senior Staff Software Engineer in IBM India Systems Development Lab, IBM India. He has over 18 years of experience in the Information Technology field. He currently works for IBM PowerVM Live Partition Mobility development team and prior to this he worked for IBM VM Recovery manager HA and DR for Power Systems solution development. He holds a Bachelor of Engineering degree in Electronics and Communication from Visvesvaraya Technological University, Belagavi in India. His areas of expertise include PowerVM virtualization, high availability and disaster recovery solutions, clustering technologies, and SAN**.**

**Adhish Kapoor** serves as a Team Lead for VMRM Recovery Manager, overseeing the High Availability/Disaster Recovery solution for Power Systems. With over 20 years of experience in the system domain, he specializes in HA/DR solutions for intricate cluster environments, as well as in the development of backup applications and SAN/NAS storage appliances. Adhish has authored technical white papers and conference articles in the HA-DR domains. He earned his Bachelor of Computer Science Engineering degree from Punjab Technical University.

**Pravin Kedia** is a CTO for Data & AI Expert Labs running the Centre of Excellence in India. Pravin has 25+ years of experience in CP4D Data Fabric, Data Science and DataOps Solutions, Big Data, Data Warehouse, Replication, Data Science and AI Architecture with international work experience in banking, telecommunication, insurance and financial markets across the world. He has an MBA in finance from The University of Kansas and a Bachelor of Electronics and Telecommunications degree from VESIT University. He has written extensively on database and replication technologies. Pravin regularly conducts data strategy, data discovery, and design workshops with C-level executives. As CTO and Solutions Architect for IBM Data and AI solutions, Pravin is responsible for technical delivery for IBM Expert Lab Services projects across the hybrid cloud product portfolio. Pravin is a PMP since 2005, a Master Inventor (Plateau 4), and serves on the IDT for Data and Governance teams. Pravin is an Open Group Certified Executive IT Specialist Thought Leader and Certified ITS Profession Champion. Pravin has received numerous rewards and has led and participated in AOT and Cloud Studies on Data and Governance.

**Youssef Largou** is the founding director of PowerM, a platinum IBM Business Partner in Morocco. He has 22 years of experience in systems, HPC, middleware, and hybrid cloud, including IBM Power, IBM Storage, IBM Spectrum®, IBM WebSphere®, IBM Db2®, IBM Cognos®, IBM WebSphere Portal, IBM MQ, ESB, IBM Cloud Pak®, SAP HANA and Red Hat OpenShift. He has worked within numerous industries with many technologies. Youssef is an IBM Champion 2020, 2021, 2022 and 2023, an IBM Redbooks Platinum Author, and has designed many reference architectures. He has been recognized as an IBM Beacon Award Finalist in Storage, Software-Defined Storage, and LinuxONE five times. He holds an engineering degree in computer science from the Ecole Nationale Supérieure des Mines de Rabat and Excecutif MBA from Emylon.

**Henry Vo** is an IBM Redbooks Project Leader with 10 years of experience in IBM. He has technical expertise in business problem solving, risk/root-cause analysis, and writing technical plans for business. He has held multiple roles at IBM, including project management, ST/FT/ETE Test, Back-End Developer, and DOL agent for NY. He is a certified IBM zOS Mainframe Practitioner, including IBM Z® System programming, Agile, and Telecommunication Development Jumpstart. Henry holds a Master of MIS (Management Information System) degree from the University of Texas in Dallas.

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

**Sanjeev Koul**, Data, Performance & Cloud Architect Expert Labs
IBM Bangalore, India

## 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 IBM Redbooks 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, IBM Redbooks  
  Dept. HYTD Mail Station P099  
  2455 South Road  
  Poughkeepsie, NY 12601-5400

## Stay connected to IBM Redbooks

- Find us on LinkedIn:

  https://www.linkedin.com/groups/2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/subscribe

- Stay current on recent Redbooks publications with RSS Feeds:

  https://www.redbooks.ibm.com/rss.html

1

# Introduction to High Availability

In an age where digital technologies underpin every facet of our lives, from critical business processes to personal communications, one aspect stands as paramount: uninterrupted access to data, applications, and services. The modern world demands nothing less than a seamless, always-on experience, and any interruption (be it due to hardware failures, software glitches, or unforeseen disasters) is met with frustration and loss. It is in this landscape that the concepts of high availability (HA) and disaster recovery (DR) emerges as a linchpin of resilience and continuity.

In this chapter, we describe the realm of HA, exploring its fundamental principles, implementation strategies, and the diverse technologies that empower it. We delve into the critical considerations that guide organizations in their pursuit of uninterrupted operations, from the tolerance for downtime to budget constraints.

In this chapter, the following topics are described:

## 1.1 Overview of High Availability and Disaster Recovery Concepts

The cost of IT downtime within the industry is influenced by various factors. Financial losses can fluctuate depending on factors such as revenue, the sector of operation, the duration of the outage, the number of individuals affected, the time of day, and more. To illustrate, businesses engaged in high-level data transactions, such as banks and online retailers, experience significantly greater losses per hour. If one of these businesses encounters an unexpected outage during peak traffic periods, the resulting damage will naturally be more substantial.

HA is the elimination of single points of failure to enable applications or workloads to operate continuously even if one or more software or hardware components, such as a server or network components fails.

When HA measures fall short, DR steps in. Typically, DR is reserved for large-scale failures, such as a complete site failure or data corruption due to cyber attacks. DR often involves manual processes, due to the high stakes involved. It involves straightforward actions like restoring from a backup or executing a site failover, but it can also become highly intricate, including analyzing the database log to maintain logical consistency and deciding individually whether to apply particular database operations or not.

## 1.2 High Availability versus Disaster Recovery

While HA and DR are often used interchangeably, they are actually two different concepts, with significant differences. This section describes both concepts and shows how they are different in scope and implementation.

DR typically involves an unforeseen incident necessitating the restoration of either an application or the entire computing infrastructure at a remote location. In DR scenarios, the expected recovery times are typically measured in hours or even days, rather than minutes.

In contrast, HA can address both planned and unplanned downtime. HA primarily involves automated, rapid, intra-site failover procedures. Its core objective is the elimination of single points of failure, which necessitate redundant components. Most systems implement three types of redundancy: hardware, software, and environmental.

For planned downtime, such as hardware or software maintenance, coordination with end users is performed, and the downtime may last for minutes or hours. With an HA setup that features a primary and secondary database server, for instance, any downtime caused by a firmware upgrade on the primary database server can be minimized by switching over to the secondary server.

Near Zero Downtime planning requires a deliberately designed HA infrastructure strategy. In cases of unplanned downtime – such as a malfunctioning hardware component or a corrupted database – an automatic failover to a secondary server takes place with minimal impact to end users.

The design of HA within a server's components, the infrastructure architecture, and software all play crucial roles in reducing unplanned downtime to a minimum. Implementing HA in an infrastructure comes with a cost. It is essential to weigh this cost against the expenses associated with downtime to justify the expense of preventing or minimizing such disruptions.

## 1.2.1 High availability

An HA system or component is specifically designed to be continuously operational for long periods of time. The HA system is built to eliminate any single point of failure by providing redundant components in the software, hardware, and applications.

Hardware redundancy means that the hardware design ensures that all critical hardware components in the system have one or more backups, ensuring that a failure of an individual component does not interrupt the operation of the system. This includes server hardware, networking components and storage. Hardware redundancy is critical for modern business data systems.

HA hardware is only one part of an HA system. Software redundancy is also required to ensure that the business applications can tolerate the failure of the system where they are running. This is usually done using clustering technologies that spread the database and application code across multiple redundant servers. If an outage of short duration can be tolerated or if your software does not support clustering capability, a simpler infrastructure utilizing VM Restart based technology can be used to quickly restart the business applications on backup hardware in a quick and efficient manner.

## 1.2.2 Disaster recovery

There are some situations that cannot be addressed by an HA system. These situations are generally the result of some incident that causes a full site failure such as fire, flooding or other natural disaster that would require the movement of your business application to another physical location, usually to another data center in a different geographic region. DR is the process that is used when your HA solution cannot continue to operate.

DR involves a set of policies and procedure to enable recovery of your business applications on an infrastructure in a different location that is unaffected by whatever disaster forces you to move operations. This must include planning for not only servers, storage, and local networking components in the new location, but also needs to consider user access to the new systems and personnel to operate the systems in the new location. This is critical to be able to continue your business operations in case of a site failure, natural disaster, or planned outage to your data center.

When you are planning and setting up your DR location it is important to understand your business requirements. The cost and complexity of your DR plan is dependent on a couple of important objectives that you need to be able to assign to your different business applications. These two objectives are Recovery Time Objective (RTO) and Recovery Point Objective (RPO).

### Recovery Time Objective

The RTO describes the maximum amount of time that a system can be down. This is the length of time from when the outage starts until the application or function is completely restored to operation.

The RTO should be determined for each business function, based on the importance of that function to your business. For some business functions, RTO can be measured in hours or even days, but for critical functions the RTO could be measured in minutes or even seconds.

### Recovery Point Objective

The RPO defines the amount of data loss, measured in time, your business application can sustain. The RPO requirement is critical to understand for each business function, as it directly affects the design and cost of your DR solution.

By definition, DR solutions require replication of data between two locations that are geographically separated. To ensure that both locations are not affected by the same event, this separation is often thousands of miles. The smaller your RTO, the more complex your DR solutions need to be to achieve that goal, and this extra complexity leads to increased costs.

HA and DR are targeted to solve the same problem: ensuring your system is up and running and limiting the loss of data caused by any failure. You can characterize the differences between HA and DR as:

- HA is intended to handle issues while the system is running:
  - Redundant components are used to recover from specific hardware or software failures.
  - The time required to recover your application depends on your RPO/RTO requirements. Using cluster technology, recovery can occur with little or no user impact. For applications with less restrictive RPO/RTO requirements, virtual machine (VM) recovery technologies can restart the application in minutes on other available hardware.
  - Software licensing costs need to be considered – you may need to acquire additional licenses depending on the specific supplier.
- DR handles problems caused by system failure or infrastructure failure that cannot be addressed by your HA solution.
  - Recovery is made to a different physical location, often geographically distant from your primary location. There will almost always be an outage required. The length of the outage is dependent on many factors.
  - The DR location hardware can often be used for other purposes (for example, testing and development) during normal operations and during a DR situation those functions will be displaced by production workloads until the primary site failure can be resolved.
  - Recovery back to production generally requires an additional outage.

### 1.2.3 Service-level agreements

Service level agreements (SLAs) represent contractual agreements between the service provider and the service user, typically entailing penalties for the provider if SLA terms are breached, often in the form of financial compensation. SLAs primarily pertain to instances of unplanned service disruptions, as opposed to planned ones.

The higher the guaranteed percentage of uptime specified in the SLA, the less time the service is allowed to be unavailable. Each additional “9” following the decimal point signifies a substantial reduction in potential downtime. For example, transitioning from 3 “9s” (99.9%) to 5 “9s” (99.999%) represents a variance of 8 hours, 40 minutes, and 41 seconds in downtime. For service providers, adhering to higher SLA availability levels necessitates significant investments in and architectural enhancements to their IT infrastructure to enhance resilience against failures. Resilience manifests in various forms.

For instance, servers like IBM Power Servers incorporate considerable built-in redundancy, enabling them to claim a hardware availability of five nines (99.999%). Similarly, IBM FlashSystem® storage boasts six nines (99.9999%) of availability. However, if the switch responsible for connecting the fiber cables between the server and storage guarantees only three nines (99.9%) availability, it becomes the weakest link, determining the overall infrastructure's availability. To address this, many clients opt to install a secondary switch along an entirely distinct fiber channel path connecting the server and storage.

Availability is defined by the SLA. Data needs to be gathered on the systems to show when a system is available and the length of any unplanned outages to determine if the SLA has been met.
Table 1-1 maps the amount of outage time allowed per year for a system to meet each availability level (from one nine to six nines). As can be seen by the values in the table, achieving availability numbers of four nines or higher requires a high level of automation and planning.

*Table 1-1 Typical SLAs*

| Number of nines | % Uptime | Maximum Annual downtime |
|---|---|---|
| Six (6) | 99.9999 | 32 seconds |
| Five (5) | 99.999 | 5 minutes 16 seconds |
| Four (4) | 99.99 | 52 minutes 36 seconds |
| Three (3) | 99.9 | 8 hours 45 minutes 57 seconds |
| Two (2) | 99.0 | 3 days 15 hours 39 minutes 29 seconds |
| One (1) | 90.0 | 36.5 days |

## 1.2.4 HA/DR strategies – Eliminating single points of failure

Every element within an IT infrastructure carries a certain, albeit low, probability of experiencing a failure. Consequently, the key to achieving HA revolves around redundancy, encompassing hardware redundancy, network redundancy, and redundancy at the data center level.

Regarding data center redundancy, the depicted configuration in Figure 1-1 showcases a three-site architecture, comprised of two on-premise sites managed by the client and an additional failover site housing servers and storage systems within the IBM Cloud. Beyond replicating the data centers, it is crucial to address potential single points of failure at the site level, such as power supplies or network connections.

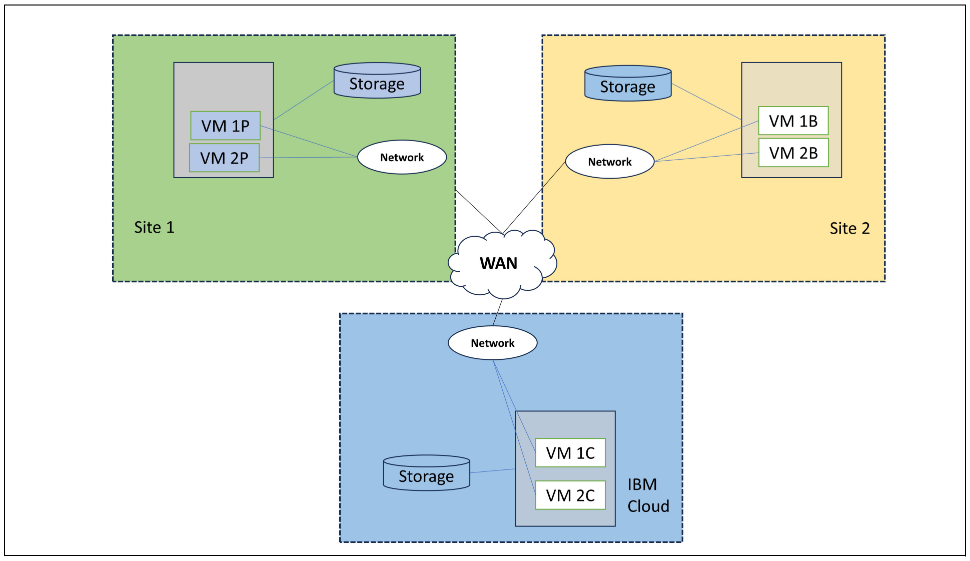


*Figure 1-1 HA and DR configuration example*

In addition to redundant hardware and redundant network connections, you also need to plan for redundant power. For example, using an Uninterrupted Power Supply device (UPS) can effectively balance power supplied from two different vendors, backed up by an onsite battery backup and emergency generators.

In terms of hardware redundancy, IBM Power offers multiple layers of redundant server components. These include redundant power supplies, fans, enterprise-grade error-correcting memory, fully redundant network adapters, and spare CPUs that can be activated in the event of a failure or for load-balancing purposes. Disk storage systems employ striping and mirroring techniques to ensure redundancy, facilitating automatic recovery from disk failures. All network components, including switches, are designed with full redundancy in mind.

In scenarios where an entire server or storage system encounters a failure, dedicated standby hosts are employed for seamless failover. Leveraging the capabilities of PowerVM, these standby hosts do not remain idle but can serve as development or test LPARs while simultaneously replicating production data from the primary server. In the event of a failover, the development and test LPARs are paused, allowing the production workload to seamlessly continue running on the standby hosts, virtually unaffected by the incident.

## 1.3 HA and DR solutions for IBM Power systems

IBM Power systems are one of the most reliable platforms in the industry and are designed to meet the requirements of the most critical data-intensive workloads. For the 15th straight year, IBM Power Servers achieved the highest server reliability ranking in the ITIC 2023 Global Server Hardware and Server OS Reliability survey[1].

IBM Power and IBM Power Virtual Server (an IBM Cloud offering based on IBM Power systems) provide an entire product line of HA and DR solutions. Figure 1-2 provides an overview of the IBM Power HA/DR capabilities on IBM Power systems.

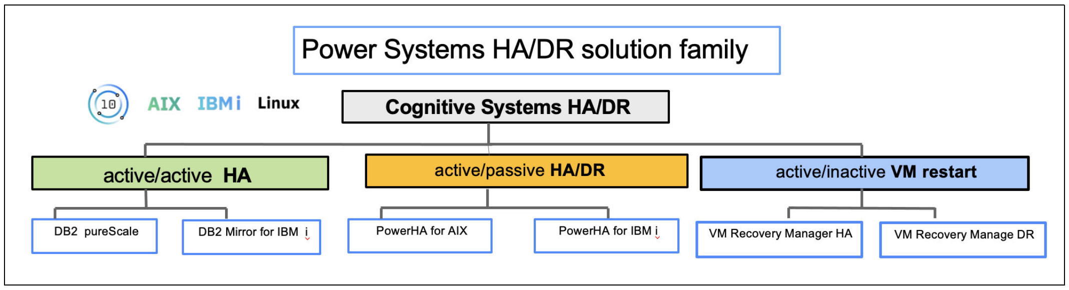


*Figure 1-2 IBM Power Systems virtual server HA/DR solution family.*

The IBM PowerHA SystemMirror® family of solutions is optimized for mission-critical applications where the total annual downtime for both planned and unplanned outages is zero or near-zero. The PowerHA System mirror product line covers all outage types, both software and hardware. There is at least one active OS on each of the nodes in the cluster, which enables software updates on a system other than the production node. PowerHA SystemMirror covers both data center and multi-site configurations. To drive total outage time for both planned and unplanned events to near zero, this solution is the one to deploy. For more information, see IBM PowerHA SystemMirror.

[1] 2023 ITIC Reports & Surveys

Virtual Machine Recovery Manager (VMRM) solutions can be best understood by first understanding Live Partition Mobility (LPM). A set of logical partitions (LPARs) is virtualized using IBM PowerVM and Virtual I/O Server (VIOS) to enable a partition to be moved for a firmware or hardware maintenance event by using LPM. If that VM fails, it can be restarted on another server in the cluster. For DR operations, those same VMs are replicated using storage area network (SAN) storage at the secondary location. For more information, see IBM VM Recovery Manager for Power.

The active-active solutions are IBM DB2® pureScale®, which supports AIX, and IBM DB2 Mirror for IBM i. In both cases, the solutions are classified as active-active, but they are achieved using different approaches. Db2 pureScale provides an active/active solution using a shared DB2 cluster configuration with distributed lock management to enable multiple application servers to simultaneously access the shared database. In Db2 pureScale, the active-active capability is first at the DB2 layer, where there are two copies that are replicated bi-directionally and synchronously. The application servers can be configured active-active or active-passive. Table 1-2 highlights the differences between IBM Power HA/DR solutions.

For more information, see *IBM Power Systems High Availability and Disaster Recovery Updates: Planning for a Multicloud Environment*, REDP-5663.

*Table 1-2 HA topology classification*

| **Technology** | **Active/Active Clustering** | **Active/Passive Clustering** | **Active/Inactive Clustering** |
|---|---|---|---|
| Definition | Application clustering: applications in the cluster have simultaneous access to production data. Therefore, there is no application restart upon a node outage. Certain types enable read-only access from secondary nodes. | OS clustering: one OS in the cluster has access to the production data. Multiple active OS instances on all nodes in the cluster. Application is restarted on a secondary node upon outage of a production node. | VM Clustering: one VM in a cluster pair has access to the data, one logical OS, and two physical copies. OS and applications must be restarted on a secondary node upon a primary node outage event. LPM enables the VM to be moved non-disruptively for a planned outage event. |
| Outage Types | ▸ SW, HW, HA, planned, unplanned<br>▸ RTO 0, limited distance | ▸ SW,HW,HA,DR, planned, unplanned<br>▸ RTO>0, multi-site | ▸ HW,HA,DR, planned, unplanned<br>▸ RTO>0, multi-site |
| OS integration | Inside the OS | Inside the OS | OS agnostic |
| RPO | Sync mode only | Sync/Async | Sync/Async |
| RTO | Zero | Fast (minutes) | Fast Enough (VM Reboot) |
| Licensing | N+N[a] | N+1 licensing | N+0 licensing |
| IBM Solution | DB2 pureScale, DB2 Mirror | PowerHA, Red Hat HA, Linux HA | VMR HA, LPM, VMR DR |

a. N = the number of licensed processor cores on each system in the cluster.

## 1.4 HA and DR options for Linux on IBM Power

For organizations leveraging Linux on IBM Power systems, the pursuit of HA and DR options takes center stage. This section explores the vital strategies and tools available to safeguard Linux-based workloads on IBM Power, delving into the key concepts, components, and real-world applications that underpin these essential practices.

### 1.4.1 VM recovery tools

The following are the key VM recovery tools available for Linux on IBM Power Systems:

- **PowerVM** is the virtualization technology for IBM Power systems. It provides the foundation for creating and managing virtualized environments, including VMs.
- **Live partition mobility** is a feature included in the PowerVM Enterprise Edition hardware offering that enables the relocation of LPARs running AIX, IBM i, and Linux from one system to another. This migration process encompasses the entirety of the system environment, encompassing processor status, memory, attached virtual devices, and active user connections.
  - With active partition migration, also known as Live Partition Mobility, you have the capability to move AIX, IBM i, and Linux logical partitions, including their running operating systems and applications, from one system to another. Importantly, this migration can be executed without the need to halt or shut down the logical partition or the applications it hosts.
  - In the case of inactive partition migration, also known as cold partition mobility, you can transfer a powered-off AIX, IBM i, or Linux logical partition from one system to another.
  - To facilitate these migrations, the Hardware Management Console (HMC) serves as a valuable tool, allowing you to efficiently move both active and inactive logical partitions from one server to another.
  - The HMC is programmed to migrate the most recently activated profile. Consequently, an inactive LPAR that has never been activated cannot undergo migration. For cases of inactive partition mobility, you have the option to either choose the partition's state as defined in the hypervisor or opt for the configuration data specified in the last activated profile on the source server.
  - Partition mobility offers enhanced systems management flexibility and is explicitly engineered to enhance system uptime with the following advantages:
    - Planned outages for hardware or firmware maintenance can be circumvented by migrating logical partitions to an alternative server before conducting the necessary maintenance. Partition mobility proves invaluable in these situations as it permits the seamless adjustment of scheduled maintenance activities.
    - Downtime related to server upgrades can be entirely avoided by migrating logical partitions to an alternate server before proceeding with the upgrade process. This allows for uninterrupted operations.
    - In anticipation of potential server failures, LPARs can be preemptively migrated to another server, thus mitigating the risk of unscheduled outages. Partition mobility serves as a preemptive measure against such incidents.
    - The consolidation of workloads, typically dispersed across several underutilized servers, onto a single, more robust server can be achieved through partition mobility.
    - Workloads can be dynamically relocated between servers to optimize resource utilization and workload performance within the computing environment. With active partition mobility, this can be achieved with minimal disruption to ongoing operations.

**Note:** Partition mobility, while delivering numerous advantages, does not encompass the following functionalities:

- Automatic workload balancing is not a feature of partition mobility.
- Partition mobility does not serve as a conduit for introducing new functionalities. To harness new features, LPARs may require restarts and, in some cases, re-installation.

- The VM Recovery Manager HA solution implements recovery of the VMs based on the VM restart technology. The VM restart technology relies on an out-of-band monitoring and management component that restarts the VMs on another server when the host infrastructure fails. The VM restart technology is different from the conventional cluster-based technology that deploys redundant hardware and software components for a near real-time failover operation when a component fails. The VM Recovery Manager HA solution is ideal to ensure HA for many VMs when the workload on those VMs can tolerate short interruptions of service while the VM is restarted. Because the VM Recovery Manager HA solution is based on the VM restart technology instead of clustering technology, it is easier to manage, as clustering introducing additional complexities.

Figure 1-3 shows the architecture of the VM Recovery Manager HA solution. A set of hosts is grouped to be backup for the others. When failures are detected, VMs are relocated and restarted on other healthy hosts within the group.

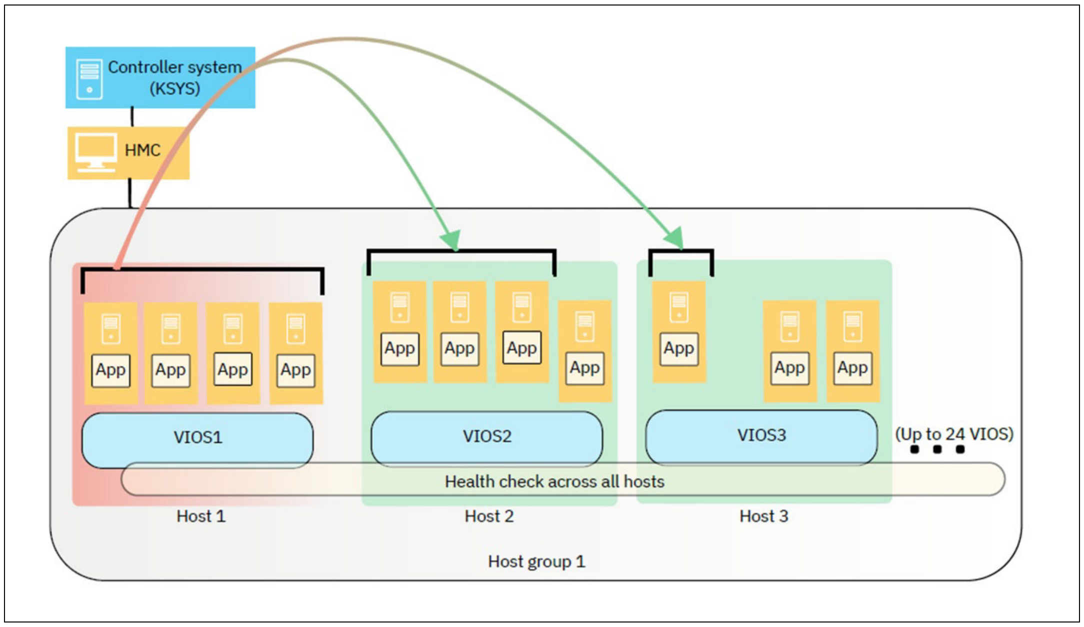


*Figure 1-3 VM Recovery Manager HA solution architecture*

## 1.4.2 Cluster solutions

Clustering is a strategic approach to ensuring the continuous availability of critical IT systems and services. It involves the use of multiple interconnected servers or nodes that work together to minimize downtime and protect against hardware failures, software glitches, or disasters.

### Key concepts and components of clustering

The following are the key concepts and components of clustering:

- **Nodes**: Clustering typically involves multiple server nodes. These nodes can be physical servers or virtual machines. Each node is equipped to run the same applications or services.
- **Load balancing**: Load balancing distributes incoming requests or workloads evenly across all nodes within the cluster. This ensures that no single node is overwhelmed with traffic, improving system performance and reliability.
- **Redundancy**: Redundancy provides replicated critical components, such as servers, storage, and network connections, to eliminate single points of failure.
- **Failover**: Failover is the ability to automatically and seamlessly switch from a failed or degraded node to a healthy one within the cluster. This ensures minimal downtime in case of a node failure.
- **Data replication**: Data replication ensures that data is copied across multiple nodes or sites to ensure data availability and integrity in the event of a disaster.
- **Quorum**: Quorum is a mechanism to prevent “split-brain” scenarios, where multiple nodes in a cluster believe they are the active node. It ensures that there is consensus among nodes before taking certain actions, such as initiating a failover.

### Types of clustering configurations

The following are common types of clustering configurations:

- **Active/Passive clustering**: In this configuration, one node (the active node) actively serves the application or service, while the other node (the passive node) remains in standby mode. If the active node fails, the passive node takes over.
- **Active/Active clustering**: In active/active clustering, both nodes actively serve the application or service simultaneously, sharing the workload. This configuration improves scalability and redundancy but requires careful load balancing.
- **N+1 clustering**: N+1 clustering involves N active nodes and one standby node. If one of the active nodes fails, the standby node takes over. This configuration balances redundancy and resource utilization.
- **N+N clustering**: N+N clustering has N active nodes running in parallel, with no standby nodes. All nodes are actively serving the application or service simultaneously. This configuration is highly redundant and scalable.
- **Three-Site Clustering**: Three-site clustering extends clustering to multiple geographically separated sites (data centers). It provides DR capabilities by replicating data and services across different locations to withstand site-level failures or natural disasters.

### Applications of clustering

The following are some of the applications of clustering:

- **Server failover**: Clustering ensures server failover in case of hardware or software failures, minimizing downtime and ensuring continuous service availability.

- **Load balancing**: Clustering improves resource utilization and responsiveness by distributing incoming traffic or workloads across multiple nodes.
- **DR**: Clustering, especially in multi-site configurations, supports DR by replicating data and services across geographically dispersed locations.
- **Application scaling**: Clustering allows for the dynamic scaling of applications and services to accommodate increased demand or resource requirements.
- **HA Databases**: Database clustering provides redundancy and failover capabilities for critical database systems, ensuring data availability and integrity.

Clustering in the context of HA and DR is a critical strategy for ensuring the resilience, redundancy, and continuous availability of IT systems and services, making it an essential component of modern business continuity planning.

## 1.5 Clustering implementations

Implementing clustering solutions, particularly in the context of HA, is a strategic endeavor that requires careful consideration of various factors. One of the critical aspects to bear in mind is striking the right balance between the tolerance for downtime and the available budget. This balance ensures that the chosen HA solution aligns with the unique needs and constraints of each business function or project. When measuring or determining acceptable downtime, it is essential to take into account several parameters:

- **Loss of data**: Assess the criticality of data loss to your organization. Some systems may require real-time data replication and failover to avoid even the slightest data loss, while others may tolerate minimal data loss.
- **Loss of an application**: Consider the impact of losing access to a specific application. For some businesses, the unavailability of a critical application even for a short duration can have severe consequences.
- **Loss of a system**: Evaluate the consequences of a server or system failure. HA solutions should address the ability to swiftly recover from hardware or software failures.
- **Loss of business location**: For multi-site organizations, contemplate the implications of losing an entire business location due to a disaster. Clustering solutions that enable site failover can mitigate this risk.
- **Loss of operations**: Gauge how a disruption in operations would affect your business. Some businesses can endure brief interruptions, while others demand continuous, uninterrupted operations.
- **Scheduled maintenance**: Plan for scheduled maintenance activities. HA solutions should facilitate seamless failover or workload migration to minimize the impact of maintenance windows.
- **Impact of downtime on user experience**: Understand how downtime affects user experience and customer satisfaction. Ensuring minimal disruption to users is a priority for many organizations.
- **Loss of business-critical applications**: Identify which applications are truly business-critical. HA solutions should prioritize these applications for rapid recovery and minimal downtime.

In practice, the choice of clustering implementation, whether it is active/passive, active/active, N+1, N+N, or a multi-site clustering setup like 3-site clustering, depends on the specific requirements of the organization. The aim is to align the HA solution with the organization's tolerance for downtime and budget constraints while safeguarding critical data, applications, and operations.

Ultimately, a robust clustering solution should be tailored to the unique needs of the business or project, taking into account the parameters mentioned above. Finding the right balance between HA requirements and budgetary considerations ensures that the chosen HA strategy optimally meets your organization's objectives for system availability and resilience.

## 1.5.1 Active/Passive clustering

Active/Passive clustering, also known as a failover cluster, is an HA configuration used to ensure the continuous availability of critical services or applications in the event of a failure. In this configuration, you have multiple servers (nodes) working together, but only one of them is actively serving the application or service at any given time, while the others remain in standby (passive) mode. The active node handles all requests and data processing, while the passive nodes are ready to take over in case the active node fails. This setup minimizes downtime and service disruption, making it an essential component of business continuity planning.

### Components of Active/Passive clustering

The following are the key components of Active/Passive clustering configuration:

- Nodes: The nodes are the servers that are participating in the cluster. In Active/Passive clustering, there is one active node and one or more passive nodes. The active node actively provides the service, while the passive nodes are on standby.
- Quorum: A quorum is a mechanism that ensures that a majority of nodes agree on the cluster's state, preventing “split-brain” scenarios where multiple nodes think they are the active node. A cluster must have a quorum before it can perform certain actions.
- Cluster Resource Manager (CRM): CRM is software that manages the cluster resources and ensures they are running on the appropriate node. Pacemaker is an example of a CRM for Linux clusters.

### Example of Active/Passive clustering

Consider a scenario where a large payroll application needs to ensure HA and handle internal user traffic as shown in Figure 1-4.

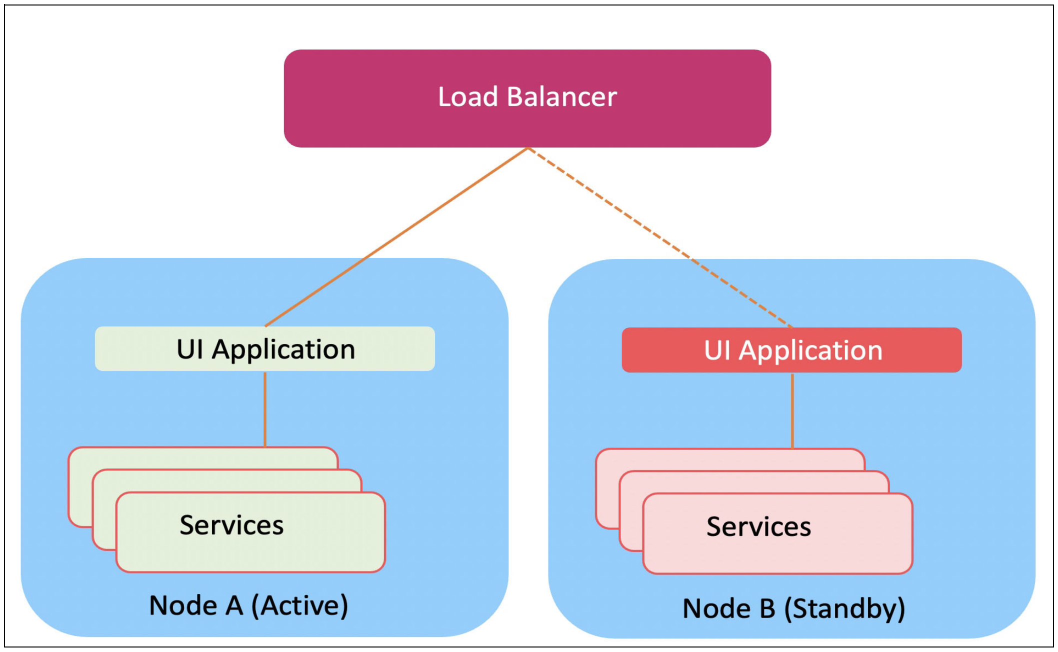


*Figure 1-4 Active/Passive clustering*

### *Scenario*

Two data centers: Each data center is equipped with a server, Node A in data center 1 and Node B in data center 2.

- Active server (Node A): This is the primary web server that actively serves web requests. It hosts the website, runs the web application, and handles incoming traffic.
- Passive server (Node B): This is the secondary web server that remains in a standby or passive state. It mirrors the configuration and content of the active server but does not actively serve web traffic.
- Load balancer: A load balancer sits in front of the active and passive servers. It distributes incoming web requests evenly between the two servers to ensure optimal resource utilization.

### *Configuration and operation*

- Normal operation (active state): In normal operation, Node A (the active server) receives and processes incoming web requests. It serves the website, runs the web application, and responds to user interactions.
- Continuous data synchronization: To maintain data consistency and redundancy, data, configurations, and database updates are continuously synchronized from Node A (active) to Node B (passive) using technologies like data replication or shared storage.
- Failover triggered by a failure: If Node A experiences a hardware failure, software crash, or any other issue that causes it to become unresponsive, the load balancer detects the failure.
- Failover to Node B (Passive State): The load balancer quickly reroutes incoming web traffic from Node A to Node B. Node B becomes the new active server, assuming the role of Node A. Users continue to access the website without interruption.
- Notification and remediation: An alert or notification is sent to system administrators to investigate and address the issue with Node A. Once the issue is resolved, administrators can manually or automatically synchronize data and configurations back to Node A.
- Load balancer adjustments: When Node A is back in a healthy state, administrators may choose to manually or automatically switch back to Node A as the active server. The load balancer adjusts accordingly to maintain a balanced traffic distribution.

## Active/Passive clustering advantages

Active/Passive clustering offers several advantages for ensuring HA of critical applications and services:

- Reduced downtime: The primary advantage of Active/Passive clustering is the reduction of downtime. In the event of a failure on the active node, the passive node takes over almost immediately, minimizing service disruption. This ensures that critical applications remain available to users, even during hardware or software failures.
- Improved reliability: Active/Passive clusters enhance the reliability of services by providing a backup node that can take over when the active node encounters problems. This redundancy increases system uptime and reduces the risk of data loss or service outages.
- Simplified maintenance: Planned maintenance, such as software updates or hardware upgrades, can be performed on the passive node without impacting service availability. When maintenance is completed on the passive node, it can be switched back to the active role, and the previously active node can undergo maintenance in a controlled manner.
- Highly predictable failover: Active/Passive clusters provide predictable failover behavior. Administrators can configure the failover policies and resource constraints to ensure that

the passive node becomes active only when specific conditions are met. This predictability is valuable for maintaining control over resource allocation and service availability.

- Cost-effectiveness: Active/Passive clustering is often more cost-effective than Active/Active clustering (where all nodes are active simultaneously) because fewer hardware resources are required for passive nodes. This makes it an attractive option for organizations with budget constraints.
- Simplified management: CRMs like Pacemaker automate the failover process, making it easier for administrators to manage high-availability configurations. They provide tools for monitoring the cluster's health, resource status, and event logging.
- Geographic redundancy: Active/Passive clusters can be set up across geographically separated data centers or locations. This geographic redundancy ensures that the service remains available even in the event of a site-wide failure, such as a natural disaster or power outage.
- Compliance and SLA fulfillment: Active/Passive clustering helps organizations meet compliance requirements and SLAs by ensuring the availability of critical services. This is especially important for industries with strict uptime requirements, such as finance, healthcare, and e-commerce.
- Risk mitigation: By providing a standby node that can take over in case of failures, Active/Passive clustering reduces the risk of costly business interruptions, reputations damage, and customer dissatisfaction associated with service downtime.

## 1.5.2 Active/Active clustering

Active/Active clustering is an HA configuration that allows multiple nodes or servers to actively serve applications or services simultaneously. In this setup, all nodes are actively processing user requests and workloads, sharing the load to improve performance, scalability, and redundancy. Active/Active clustering is commonly used in scenarios where HA and resource optimization are critical.

### Key concepts and components of Active/Active clustering

The following are the key components of Active/Active clustering configuration:

- Multiple active nodes: Active/Active clustering involves two or more active nodes, often referred to as “members” or “instances.” These nodes are equally capable of running the same applications or services concurrently.
- Load balancing: To evenly distribute incoming user requests and workloads among the active nodes, load-balancing mechanisms are used. Load balancers direct traffic to the appropriate node based on factors like server load, response times, or other defined criteria.
- Resource sharing: Active/Active clusters allow resources, such as CPU, memory, and storage, to be shared among all nodes. This sharing ensures efficient utilization of resources and prevents overloading of any single node.
- Data synchronization: For stateful applications or services that require shared data, data synchronization mechanisms are implemented to keep data consistent across all active nodes. This can involve real-time replication or shared access to a centralized database.
- Redundancy: Active/Active clustering aims to eliminate single points of failure by providing multiple active nodes. If one node fails, the remaining nodes continue to provide service, minimizing downtime.

## Example of Active/Active clustering

Consider a scenario where a large e-commerce website needs to ensure HA and handle a substantial volume of user traffic.

### *Scenario*

Two data centers: Data Center A and Data Center B, each equipped with multiple servers.

E-commerce application: The website's e-commerce application serves customers' online shopping needs.

An Active/Active scenario is illustrated in Figure 1-5.

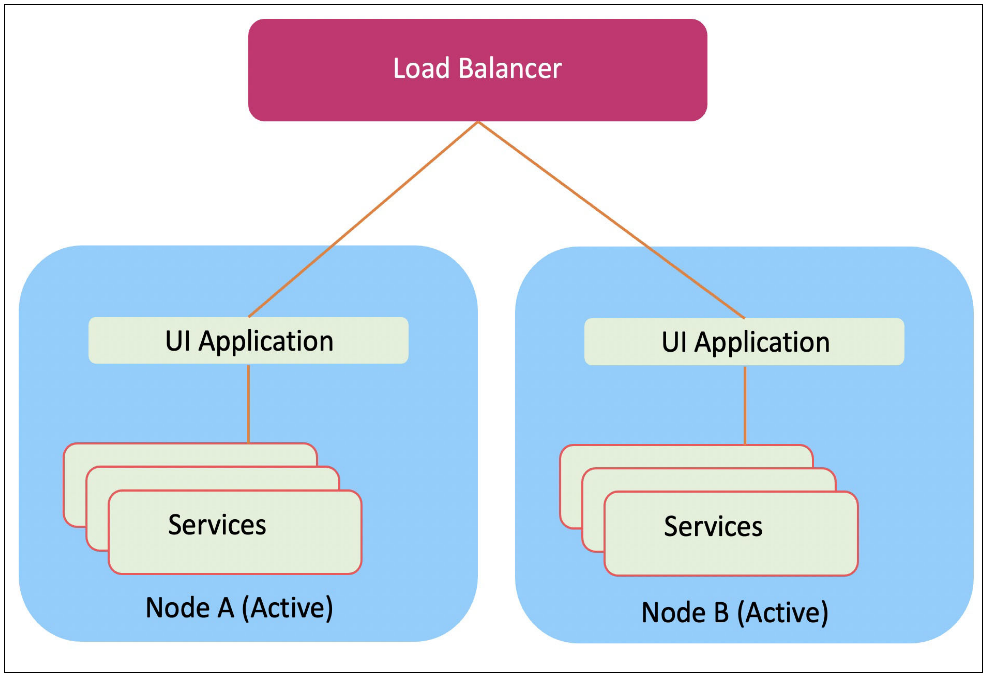


*Figure 1-5 Active/Active Clustering*

### *Configuration and operation*

- Load balancing: Incoming user requests are directed to Data Center A or Data Center B based on factors such as server load and geographic proximity. Load balancers distribute the traffic evenly across all active nodes within each data center.
- Active nodes: In each data center, there are multiple active nodes (servers) running the e-commerce application. All nodes are actively processing user requests and managing the website's database.
- Data replication: To maintain data consistency, the website's database is replicated in real-time between Data Center A and Data Center B. Changes made in one data center are immediately reflected in the other.
- Failover: If a server in Data Center A fails due to hardware or software issues, the load balancer detects the failure and redirects traffic to the remaining servers in Data Center A and Data Center B. Customers experience minimal service interruption.
- Resource sharing: During normal operation, the active nodes in both data centers share the processing load, ensuring efficient resource utilization.

### Benefits of Active/Active clustering

The following are the key benefits of Active/Active clustering configuration:

- HA: Active/Active clustering offers excellent availability by allowing all nodes to actively serve traffic. Failures on one node do not result in service downtime.
- Scalability: It provides scalability and the ability to handle increased user traffic by adding more active nodes.
- Resource efficiency: Resource sharing among nodes optimizes resource utilization and improves performance.
- Load balancing: Load balancing ensures even distribution of workloads and responsive service delivery.
- Data consistency: Real-time data synchronization guarantees data consistency and integrity.

Active/Active clustering is commonly used in scenarios where continuous availability, scalability, and resource optimization are essential, such as e-commerce websites, content delivery networks (CDNs), and large-scale cloud-based applications. It is a powerful approach to maintaining HA while efficiently utilizing hardware and network resources.

## 1.5.3 N +1 clustering

N+1 clustering is an HA configuration that provides redundancy and fault tolerance for critical applications and services by having one extra (N+1) node in standby mode compared to the number of actively functioning nodes (N). In the event of a failure on one of the active nodes, the standby (extra) node takes over, ensuring continuous service availability. This setup is commonly used in data centers and critical infrastructure to minimize downtime and maintain uninterrupted operations.

### Key components of N+1 clustering

The following are the key components of N+1 clustering configuration:

- N active nodes: These are the primary nodes that actively serve the application or service. They handle user requests, process data, and perform the core functions of the system.
- 1 standby node: The standby node is kept in reserve, not actively participating in processing requests but ready to take over in case of a failure in any of the active nodes. This ensures redundancy and fault tolerance.
- Failover mechanism: The cluster is equipped with a failover mechanism that automatically detects when an active node fails and initiates the transition of services to the standby node. This mechanism may involve monitoring for node health, resource availability, and network connectivity.

### Example of N+1 clustering

Let us consider an example of an N+1 cluster in the context of a web application hosted in a data center.

#### *Scenario*

- N = 2 Active Nodes (Node A and Node B): These nodes actively host the web application, serving user requests and processing data.
- 1 standby node (Node S): This node is in standby mode and is not actively serving requests. It has the same configuration as Node A and Node B.

*Configuration and operation*

- Normal operation: During normal operation, Node A and Node B actively handle incoming web requests. Load-balancing mechanisms distribute traffic between them, ensuring optimal performance. Node S remains in standby mode, monitoring the health of Node A and Node B.
- Failure detection: If Node A encounters a hardware failure, becomes unresponsive, or experiences a critical software issue, the failover mechanism detects the problem.
- Failover process: The failover mechanism triggers a failover process, directing incoming traffic away from the failed Node A. Node S, the standby node, is brought into active mode, assuming the responsibilities of Node A. It now starts processing web requests.
- Service continuity: Users accessing the web application may experience a brief interruption or delay during the failover process, but the application remains accessible. Node B continues to operate as an active node, serving web requests alongside Node S.
- Recovery and maintenance: After Node A's issue is resolved or maintenance is completed, it can be brought back into the cluster as a standby node. The cluster can be reconfigured to maintain N+1 redundancy, with Node A and Node B active, and Node S back in standby mode.

#### Benefits of N+1 clustering

The following are the key benefits of N+1 clustering configuration:

- HA: N+1 clustering ensures continuous service availability even in the face of node failures, hardware issues, or software problems on active nodes.
- Fault tolerance: It enhances fault tolerance by providing a ready-to-use standby node, reducing the risk of service disruptions.
- Scalability: Additional standby nodes can be added for further redundancy and capacity, making it suitable for scaling as needed.
- Reliability: N+1 clustering is a reliable solution for critical applications and services where downtime is costly or unacceptable.

N+1 clustering is widely used in various industries, including finance, healthcare, telecommunications, and e-commerce, to maintain the availability of mission-critical systems and services.

### 1.5.4 N-to-1 clustering

The following are the key characteristics of N-to-1 clustering configuration:

- Redundancy: In an N-to-1 clustering configuration, "N" components (servers, nodes, or modules) work together as active units to handle the workload and serve applications.
- 1 Standby or Concentrator: The "1" component, often referred to as the "concentrator" or "controller," is a single unit that acts as a central point of management or control. It does not actively process workloads but coordinates and directs traffic among the "N" active components.
- Failover: If one of the active components fails, the standby concentrator takes over the management and routing of traffic. This ensures that even if one of the active units fails, the system can continue to operate with minimal disruption.
- Examples: N-to-1 clustering is commonly used in load balancing and traffic management scenarios, where multiple active servers distribute incoming requests, and a single concentrator handles traffic management and failover.

**Note:** The key distinction between N+1 and N-to-1 clustering lies in the role of the additional component(s). In N+1, you have "N" active components with a single standby or spare, while in N-to-1, you have "N" active components working in conjunction with a central concentrator that manages traffic and provides failover capabilities.

## 1.5.5 N+N clustering

N+N clustering is an HA configuration designed to provide redundancy and fault tolerance for critical applications and services by having an equal number of active nodes (N) working simultaneously. Unlike N+1 clustering, where there is one standby node for redundancy, N+N clustering has all N nodes actively serving the application or service at the same time. This approach aims to distribute the workload evenly across multiple nodes, improving performance and ensuring that if one node fails, the others can continue to provide services without interruption. N+N clustering is commonly used in scenarios where HA and scalability are paramount.

### Key components of N+N clustering

The following are the key components of N+N clustering configuration:

- N active nodes: These are the primary nodes that actively serve the application or service.
- All N nodes are configured identically and work concurrently to process user requests and data.

### Example of N+N clustering

Let us consider an example of N+N clustering in the context of a web application hosted in a data center as shown in Figure 1-6.

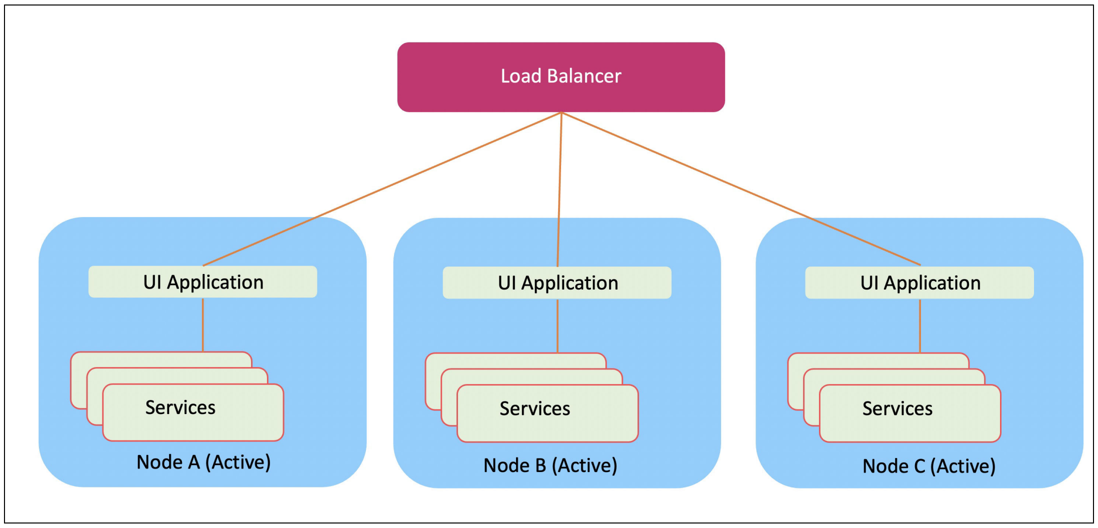


*Figure 1-6 N+N Clustering*

#### *Scenario*

- N = 3 Active Nodes (Node A, Node B, Node C): These nodes actively host the web application, serving user requests and processing data simultaneously.
- There is no standby node in this configuration.

### *Configuration and operation*

- Normal operation: During normal operation, all N nodes (Node A, Node B, and Node C) actively handle incoming web requests. Load-balancing mechanisms distribute traffic evenly across them, ensuring optimal performance and scalability.
- Failure detection: If one of the nodes, say Node B, encounters a hardware failure, software issue, or becomes unresponsive, the system detects the problem.
- Continued service: Despite the failure of Node B, the web application continues to operate without disruption. User requests are automatically routed to Nodes A and C, which are still active and operational.
- Service continuity: Users accessing the web application may not even notice the failure of Node B. The N+N clustering configuration ensures that there is no single point of failure, and the system continues to provide services reliably.
- Recovery and maintenance: After resolving the issue with Node B or performing necessary maintenance, it can be reintegrated into the cluster as an active node. This process can be repeated for any failed node.

### Benefits of N+N clustering

The following are the key benefits of N+1 clustering configuration:

- HA: N+N clustering offers HA by distributing the workload across multiple active nodes. Even if one or more nodes fail, others continue to operate, minimizing service downtime.
- Redundancy: There is no single point of failure in N+N clustering, as multiple nodes are actively serving the application or service.
- Scalability: N+N clusters are highly scalable. Additional nodes can be added to increase capacity and further distribute the workload, accommodating growth.
- Performance: N+N clustering typically results in improved performance and response times since multiple nodes share the load, making it suitable for high-demand environments.

N+N clustering is often used in scenarios where service uptime is crucial and where handling fluctuations in workload is essential. It provides a robust solution for businesses and organizations that require continuous availability and scalability for their critical applications and services.

## 1.5.6 Three-site clusters

A three-site cluster, often referred to as a multi-site or three-node clustering configuration, is an HA setup designed to provide redundancy and DR capabilities across three geographically separate locations or data centers. This configuration is suitable for organizations that require high levels of availability, data protection, and continuity of critical services, even in the face of site-level failures or natural disasters.

### Key components of three-site clustering

The following are the key components of a three-site clustering configuration:

- Three sites: In this configuration, there are three distinct physical locations or data centers, each housing a portion of the cluster nodes and resources. These sites are geographically separated to reduce the risk of simultaneous failures due to local disasters.
- Cluster nodes: Each site contains a set of cluster nodes (typically two or more) that actively serve the application or service. These nodes work together to provide redundancy and HA.

- Data replication: Data is actively replicated between the sites, ensuring that data is available at multiple locations. This replication can be synchronous (real-time) or asynchronous (with a delay) depending on the specific requirements and technologies used.
- Load balancing: Load-balancing mechanisms distribute user requests and workloads across the cluster nodes within and across the three sites.

## Example of three-site clustering

Let us consider an example of a three-site clustering configuration in the context of a financial institution that relies on continuous availability and data protection:

### *Scenario*

- Three sites: Site A, Site B, and Site C, are each located in different cities or regions.
- N Active Nodes at each site (Node A1, Node A2 at Site A; Node B1, Node B2 at Site B; Node C1, Node C2 at Site C): Each site has multiple active nodes that host the core banking application.

### *Configuration and operation*

- Normal operation: During normal operation, all nodes at each site actively handle user transactions and data processing. Data is continuously replicated and synchronized between the sites in real-time (synchronous replication) to ensure data consistency and availability.
- Load balancing and geographic redundancy: Load-balancing mechanisms distribute incoming user requests across the nodes within each site to ensure optimal performance. In addition to load balancing within a site, DNS or Global Server Load Balancers can be used to distribute traffic evenly across the three geographically separated sites, providing geographic redundancy.
- Failure and DR: In the event of a site-level failure (for example, a natural disaster or complete data center outage at Site A), the remaining two sites (B and C) continue to provide services. Failover mechanisms redirect user traffic and data access to the surviving sites, ensuring service continuity.
- Data resilience: Data replication ensures that even if one site becomes unavailable, data is accessible from the remaining sites. This redundancy protects against data loss.
- Failback and Recovery: After the failed site (for example, Site A) is restored, data is resynchronized, and it can rejoin the cluster, resuming normal operations.

## Benefits of three-site clustering

The following are the key benefits of the three-site clustering configuration:

- HA: Three-site clustering offers exceptional availability by spreading services and data across three geographically separated locations.
- DR: It provides robust DR capabilities, protecting against site-level failures or natural disasters.
- Data protection: Continuous data replication ensures data integrity and minimizes the risk of data loss.
- Geographic redundancy: Geographic dispersion of sites reduces the risk of simultaneous failures due to localized events.
- Compliance: It helps organizations meet regulatory compliance requirements, particularly in industries like finance and healthcare.

Three-site clustering is a complex and costly solution, typically reserved for mission-critical systems where even a brief downtime can result in significant financial loss or regulatory non-compliance. It is essential to carefully plan, implement, and test such configurations to ensure they meet the specific requirements of the organization.

Table 1-3 highlights the key differences between Active/Passive, N+1, N+N, and three-site clustering configurations

*Table 1-3 Clustering options*

| Aspect | Active/Passive | N+1 | N+N | Active/Active | Three-Site |
|---|---|---|---|---|---|
| Number of Active Nodes | 1 | N | N | N (all actively serving) | N (across multiple sites) |
| Number of Standby Nodes | 1 | 1 | None or optional N+1 | None | None |
| Fault Tolerance | Limited | Moderate | High | High | Exceptionally High |
| Redundancy | Standby Node Redundancy | Standby Node Redundancy | Full Node Redundancy | Full Node Redundancy | Geographic Redundancy |
| Load Balancing | No | No (standby is idle) | Yes (evenly distributed) | Yes (evenly distributed) | Yes (load balanced within sites) |
| Scalability | Limited | Moderate | High | High | High |
| Geographic Redundancy | No | No | No (typically single-site) | No (typically single-site) | Yes (across geographically separate sites) |
| Complexity | Low | Moderate | Moderate to High | Moderate to High | High (complex to configure) |
| Data Replication | Optional | Optional | Optional | Optional | Real time (synchronous or asynchronous) |
| Use Cases | Simple applications with low tolerance for downtime | Moderate tolerance for downtime, scalability needed | HA with redundancy and scalability | HA with efficient resource utilization | Mission-critical applications requiring the highest levels of availability, DR, and data protection |
| Cost | Lower (fewer hardware resources) | Moderate (additional standby node) | Moderate to High (multiple active nodes) | Moderate to High (additional hardware and load balancers) | High (multiple sites, extensive hardware, and network infrastructure) |
| Examples | Single server with a standby backup | A web server farm with one standby server | Database clusters with multiple active nodes | Large-scale e-commerce websites with efficient resource utilization | Large-scale data centers with geographically dispersed sites |

**Note:** The suitability of each clustering configuration depends on specific organizational requirements, budgets, and the level of availability needed for the applications or services in question. The choice between these configurations should be based on a thorough analysis of business needs and risk tolerance.

### 1.5.7 Linux Pacemaker

Pacemaker serves as an HA CRM, a software solution designed to operate on a group of hosts, known as a cluster of nodes, with the objective of maintaining system integrity and minimizing downtime for critical services and resources. For more information, see What is Pacemaker?.

Key features of Pacemaker include:

- Detection and recovery failure at both the node and service levels.
- Ensuring data integrity by isolating and addressing malfunctioning nodes.
- Supporting single or multiple nodes within a cluster.
- Providing compatibility with various resource interface standards, enabling the clustering of virtually any scriptable component.
- Offering optional support for shared storage.
- Adapting to a wide range of redundancy configurations, including active/passive and N+1 setups.
- Automatically replicating configurations, allowing updates from any cluster node.
- Enabling the specification of cluster-wide relationships between services, including ordering, collocation, and anti-collocation constraints.
- Supporting advanced service types such as clones (services active on multiple nodes), promotable clones (clones with dual roles), and containerized services.
- Offering a unified and scriptable set of cluster management tools.

Pacemaker is developed and maintained by the ClusterLabs community. This collaborative effort unites numerous open-source projects related to HA into a comprehensive cluster solution suitable for deployments of varying sizes. Comprising components like Corosync, Pacemaker, DRBD, ScanCore, and others, the ClusterLabs stack has been instrumental in detecting and recovering from hardware and application-level failures in production clusters since 1999. Its flexibility extends to accommodating virtually any imaginable redundancy configuration.

For more information, see Chapter 2, “Pacemaker Overview” on page 25.

### 1.5.8 HA with Red Hat Enterprise Linux Add-On

The Red Hat Enterprise Linux Add-On offers the flexibility to establish managed, highly available clusters comprising Red Hat Enterprise Linux (RHEL) servers. The Add-On boasts extensive configurability, catering to a wide range of applications, whether they are off-the-shelf or custom-built. Key components of RHEL High Availability include:

- The Pacemaker CRM.
- Network management via Corosync and Kronosnet, facilitating seamless communication among cluster nodes.
- Multisite and stretch cluster support through Booth, QDevice, and QNetd.

- Management options including a Pacemaker command-line interface (pcs) and GUI (pcsd).
- A comprehensive library of resource agents for commonly used applications, alongside the ability to create custom resource agents.
- Fencing agents for bare metal servers, virtual machines, and cloud platforms.

RHEL is well-suited for a diverse array of clustering workloads, encompassing configurations such as active/passive, active/active, and primary/secondary. Its automatic service restart and failover mechanisms contribute to maintaining HA by swiftly transitioning failing services and their dependencies to functional nodes.

This HA solution seamlessly integrates with popular applications like SAP HANA, Apache, PostgreSQL, and DB2. Furthermore, custom applications can be effortlessly managed within the RHEL High Availability framework, either by utilizing their systemd startup scripts or by creating straightforward custom resource agents.

### 1.5.9 SUSE Linux Enterprise High Availability Extension

SUSE Linux Enterprise High Availability Extension is a powerful solution designed to enhance the reliability and availability of critical services and applications in enterprise environments. At the heart of this extension lies Pacemaker, a robust open-source CRM that plays a pivotal role in orchestrating HA clustering. For more information, see SUSE Linux Enterprise High Availability 15 SP1 (unsupported).

## 1.6 Linux Pacemaker vs VM Recovery Manager HA

The choice between Linux Pacemaker and IBM Power VM Recovery Manager depends on the specific needs of your environment. Linux Pacemaker is a versatile open-source solution suitable for Linux-based HA clustering across different hardware platforms. In contrast, IBM Power VM Recovery Manager is tailored for IBM Power Systems, focusing on VM-level HA and integration with PowerVM virtualization features. The decision should consider your infrastructure, budget, and the level of HA required for your workloads.

Table 1-4 highlights comparison between Pacemaker and VM Recovery Manager HA.

*Table 1-4 Linux Pacemaker vs VMRM HA.*

| **Solution** | **Pacemaker for IBM Power Systems** | **VM Recovery Manager – HA for IBM Power Systems** |
|---|---|---|
| Base technology | Linux OS | IBM PowerVM & shared storage |
| Automated failover | Yes | Yes |
| Ideal for DR | No | No |
| Ideal for HA | Yes | Yes |
| Ease of Management | Easy | Advanced |
| Reuse secondary for non-production workload | Partly | Partly |
| DR Rehearsal feature for non-disruptive DR testing | No | No |
| Cloud support | Yes | No |

| Solution | Pacemaker for IBM Power Systems | VM Recovery Manager – HA for IBM Power Systems |
|---|---|---|
| License cost | Part of Linux OS | Low cost |

## 1.7 Software licensing challenges in HA setups

When it comes to setting up a cluster scenario for HA, it is crucial to exercise caution, especially when it comes to software licensing considerations. While HA clustering can provide resilience and minimize downtime, it may have implications for software licensing costs. Some software licenses are structured in a way that charges organizations based on the number of installations, even in an active/passive scenario. Here are a few examples to illustrate this point:

- Many database management system (DBMS) vendors charge licensing fees based on the number of installations or nodes in a cluster, regardless of whether those nodes are actively processing transactions or in a standby (passive) state. For instance, if you have an active/passive cluster for a database server, you may be required to pay licensing fees for both the active and passive nodes, effectively doubling the licensing costs.
- Middleware and application server software, commonly used for hosting web applications and services, may also have licensing models that consider each node in a cluster as a separate installation. This means that even if one server is actively serving requests while the other is on standby, you could still be charged for both.
- Some operating system vendors may have licensing policies that require licensing for each node, irrespective of whether they are actively serving workloads or serving as a failover in an active/passive cluster. This can significantly impact licensing costs for organizations running HA clusters.
- Various third-party software components, including monitoring tools, security solutions, and backup software, may have licensing models that aren't cluster-aware. As a result, organizations need to carefully review their licensing agreements to ensure compliance.

To address these licensing challenges in HA scenarios, organizations should take the following steps:

- Engage with software vendors to understand their licensing policies specifically regarding clustered environments. Seek clarification on whether passive nodes are subject to licensing fees.
- Thoroughly review software licensing agreements to identify any clauses or terms related to clustering and passive nodes. Seek legal counsel if necessary.
- Explore alternative software options that have more cluster-friendly licensing models, such as licensing based on active nodes only.
- Depending on licensing costs, consider optimizing your cluster configuration to minimize passive nodes or explore active/active clustering where both nodes are actively serving traffic to avoid additional licensing fees.

2

# Pacemaker Overview

This chapter provides a more in-depth view of Pacemaker, describing how Pacemaker can be used to provide high availability for your Linux workloads. We discuss the basic components of a Pacemaker installation and help you understand how they work together to ensure that your application environment is protected from component failures.

We also discuss the various implementation topologies that can be used from the simplest active failover cluster to multi site solutions that can provide application failover not only within a single location, but across multiple sites to provide a DR solution.

This chapter contains the following topics:

- 2.1, “Architecture” on page 26.
- 2.2, “Pacemaker clustering options” on page 29.
- 2.3, “Failover policies” on page 34.

## 2.1 Architecture

Pacemaker is an HA CRM software that runs on a set of hosts to create a cluster to preserve integrity and minimize downtime of desired services or resources. Pacemaker achieves maximum availability for the cluster resources by detecting and recovering from node and resource level failures – making use of the messaging and membership capabilities provided for the preferred cluster infrastructure using CoroSync.

### CoroSync

CoroSync provides cluster infrastructure functionality. CoroSync provides messaging and membership functionality and maintains the quorum information. This feature has been utilized by Pacemaker to provide a high availability solution.

### What is Quorum?

To maintain cluster integrity and availability, cluster systems use a concept known as quorum to prevent data corruption and loss. A cluster has quorum when more than half of the cluster nodes are online. To mitigate the chance of data corruption due to failure, Pacemaker by default stops all resources if the cluster does not have quorum. The term "split-brain" generally describes a problematic condition that HA cluster solutions must avoid or address when members of a cluster lose contact with each other. When cluster members are unable to communicate, then they may be at risk of performing conflicting activities because they cannot coordinate with each other. For more information, see Exploring Concepts of RHEL High Availability Clusters - Quorum.

Quorum is established using a voting system. When a cluster node does not function as it should or loses communication with the rest of the cluster, the majority of working nodes can vote to isolate and if needed, fence the node for servicing.

### What is Fencing?

Fencing is the mechanism that the cluster uses to resolve issues and failures. Fencing ensures that any problematic cluster node will be cut off quickly and the remaining nodes in the cluster can take over the cluster services, ensuring a more resilient and stable cluster.

For example, if communication with a single node in the cluster fails, then other nodes in the cluster must be able to restrict or release access to resources that the failed cluster node may have access to. This cannot be accomplished by contacting the cluster node itself as the cluster node may not be responsive. Instead, you must provide an external method, which is called fencing with a fence agent. A fence device is an external device that can be used by the cluster to restrict access to shared resources by an errant node, often by issuing a hard reboot on the cluster node.

### Fence device types

A fence agent (or device) is an external device that can be used by the cluster to restrict access to shared resources by an errant node (or hard reboot the cluster node). The two most common types of fencing are:

- **Power fence agents:** The cluster software logs in via Telnet, Secure Shell (SSH), or Simple Network Management Protocol (SNMP) to the device such as an Automatic Transfer Switch (ATS), Dell Remote Access Controller (DRAC), HP Integrated Lights-Out (iLO), IBM Remote Supervisor Adapter (RSA), or similar device and turns off (and optionally on) the power for the cluster node. This method will execute a hard “off” action.

- **I/O fence agents:** The cluster software logs in to a fiber channel switch via Telnet or SSH and disables the ports for that node, thereby cutting off its access to shared storage. This method requires that an administrator manually reboot or shut down the errant node to recover it, and log in to the switch interface to re-enable the appropriate ports.

For more information, see Fencing in a Red Hat High Availability Cluster.

## 2.1.1 Cluster architecture

The following provides a high-level overview of the terminology used in HA cluster architecture:

- **Node:** One of a set of hosts or computers that work together to perform a task as a member of the cluster. The maximum number of nodes that can be members of  a RHEL High Availability cluster is 32. For more information, see Support Policies for RHEL High Availability Clusters - Membership and cluster size.
- **Resources:** The services that need to be kept highly available (for example, IP Address, file system, or application service).
- **Resource group:** A resource group is a collection of resources that function together and can fail over from one node to another as a single unit.
- **Resource agents:** These are scripts or operating system components that start, stop, and monitor resources, given a set of resource parameters.
- **Fence agents:** These are scripts that execute node fencing actions, given a target and fence device parameters.
- **Cluster membership layer:** This component provides reliable messaging, membership, and quorum information about the cluster.
- **Cluster resource manager:** Pacemaker provides the brain that processes and reacts to events that occur in the cluster. These events may include nodes joining or leaving the cluster; resource events caused by failures, maintenance, or scheduled activities; and other administrative actions. To achieve the desired availability, Pacemaker may start and stop resources and fence nodes.
- **Cluster tools:** These provide various command-line and graphical (GUI) interfaces for users to interact with the cluster.

For more information, see Pacemaker Administration.

Figure 2-1 illustrates Pacemaker's architecture and main components:

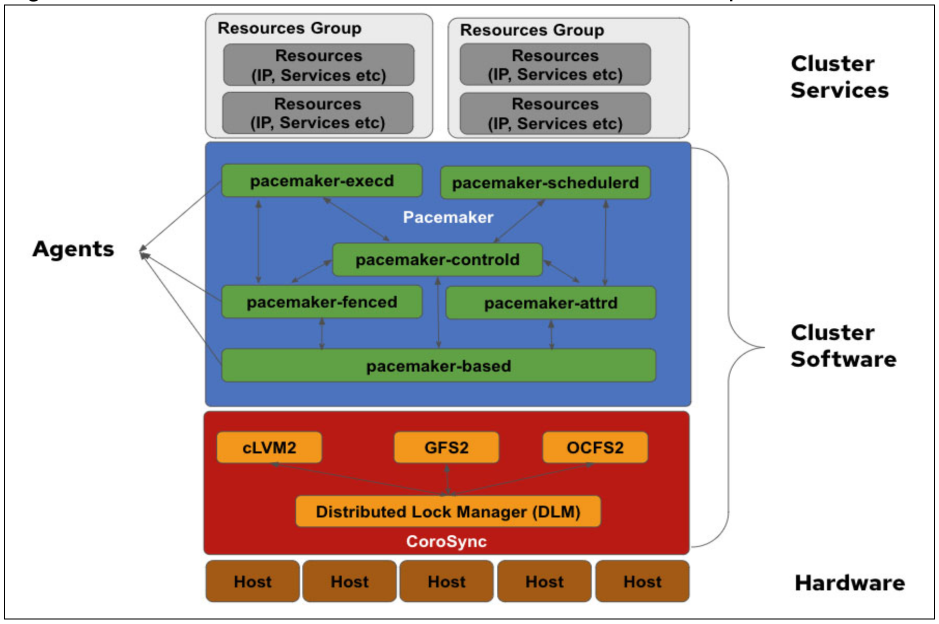


*Figure 2-1 Pacemaker architecture and components*

## 2.1.2 Pacemaker architecture and components

A cluster configured with Pacemaker comprises separate component daemons that monitor cluster membership, scripts that manage the services, and resource management subsystems that monitor the disparate resources. Pacemaker's main process (pacemakerd) spawns all the other daemons, and respawns them if they unexpectedly exit.

The following components form the Pacemaker architecture:

- Pacemaker centralizes cluster decision-making by electing one of the controller instances as the **Designated Controller (DC)**. Should the elected DC process (or the node it is on) fail, a new one is quickly established. The DC responds to cluster events by taking a current snapshot of the CIB, feeding it to the scheduler, then asking the executors (either directly on the local node, or via requests to controller peers on other nodes) and the fencer to execute any necessary actions.
- Cluster Information Base (CIB): The Pacemaker information daemon, which uses XML internally to distribute and synchronize current configuration and status information from the Designated Coordinator (DC), a node assigned by Pacemaker to store and distribute cluster state and actions by means of the CIB to all other cluster nodes.
- The CIB manager ***(pacemaker-based)*** keeps the CIB synchronized across the cluster, and handles requests to modify it.
- The attribute manager ***(pacemaker-attrd)*** maintains a database of attributes for all nodes, keeps it synchronized across the cluster, and handles requests to modify them. These attributes are usually recorded in the CIB.
- The scheduler ***(pacemaker-schedulerd)*** determines what actions are necessary to achieve the desired state of the cluster by given a snapshot of the CIB as input.
- Cluster Resource Management Daemon (**CRMd**): Pacemaker cluster resource actions are routed through this daemon. Resources managed by CRMd can be queried by client systems, moved, instantiated, and changed when needed. Each cluster node also includes a local resource manager daemon (**LRMd**) that acts as an interface between

CRMd and resources. LRMd passes commands from CRMd to agents, such as starting and stopping and relaying status information.

- The local executor ***(pacemaker-execd)*** handles requests to execute resource agents on the local cluster node, and returns the result.
- Shoot the Other Node in the Head (**STONITH**): STONITH is the Pacemaker fencing implementation. It acts as a cluster resource in Pacemaker that processes fence requests, forcefully shutting down nodes and removing them from the cluster to ensure data integrity. STONITH is configured in the CIB and can be monitored as a normal cluster resource.
- The fencer ***(pacemaker-fenced)*** handles requests to fence nodes. Given a target node, the fencer decides which cluster nodes should execute which fencing devices, and calls the necessary fencing agents (either directly, or through requests to the fencer peers on other nodes), and returns the result.
- The controller ***(pacemaker-controld)*** is Pacemaker's coordinator, maintaining a consistent view of the cluster membership and orchestrating all the other components.

For a cluster to operate, it requires an effective communications between the nodes and to manage membership of the cluster. Pacemaker utilizes CoroSync (the function with daemon of the same name) to serve the core membership and member-communication needs for high availability clusters. It is required for any High Availability Add-On capabilities to function.

The CoroSync membership and messaging functions also:

- Manage quorum rules and determination.
- Provide messaging capabilities for applications that coordinate or operate across multiple members of the cluster and thus must communicate stateful or other information between instances.
- Use the kronosnet library as its network transport to provide multiple redundant links and automatic failover.

Most managed services (applications) are not themselves, ***cluster-aware***. However, many popular open-source cluster file systems make use of a common ***Distributed Lock Manager (DLM)***, which makes direct use of CoroSync for its messaging and membership capabilities and Pacemaker for the ability to fence nodes.

For more information, see Pacemaker architecture components.

## 2.2 Pacemaker clustering options

Pacemaker supports practically any node redundancy configuration including Active/Active, Active/Passive, N+1, N+M, N-to-1, and N-to-N. The most common size for an HA cluster is a two-node cluster, since that is the minimum required to provide redundancy, but many clusters consist of many more, sometimes dozens of nodes.

An HA cluster is categorized into one of the following models[1]:

- Active/active: Traffic intended for the failed node is either passed onto an existing node or load balanced across the remaining nodes. This is usually only possible when the nodes use a homogeneous software configuration.

[1] Node configurations

- Active/passive: Provides a fully redundant instance of each node, which is only brought online when its associated primary node fails. This configuration typically requires the most extra hardware.
- N+1: Provides a single extra node that is brought online to take over the role of the node that has failed. In the case of heterogeneous software configuration on each primary node, the extra node must be universally capable of assuming any of the roles of the primary nodes it is responsible for. This normally refers to clusters that have multiple services running simultaneously; in the single service case, this degenerates to active/passive.
- N+M: In cases where a single cluster is managing many services, having only one dedicated failover node might not offer sufficient redundancy. In such cases, more than one (M) standby server is included and available. The number of standby servers is a trade-off between cost and reliability requirements.
- N-to-1: Allows the failover standby node to become the active one temporarily, until the original node can be restored or brought back online, at which point the services or instances must be failed-back to it in order to restore HA.
- N-to-N: A combination of active/active and N+M clusters, N to N clusters redistribute the services, instances or connections from the failed node among the remaining active nodes, thus eliminating (as with active/active) the need for a 'standby' node, but introducing a need for extra capacity on all active nodes.

All of these HA cluster models consist of multiple physical resources (compute nodes, share storage, network, fence device) and logical resources (node, resources, resource group) that could be collocated at one geographical location or data center (DC), or could be distributed across multiple geographical locations or data centers (DCs).

### 2.2.1 Single site cluster

The single site cluster is a setup in which all cluster members and the associated resources are in the same physical location or data center (DC), connected by a local area network.

Figure 2-2 show an example of a single site Pacemaker cluster architecture for database with replication. The database in this setup would be configured so that data is synchronized from the primary to secondary node.

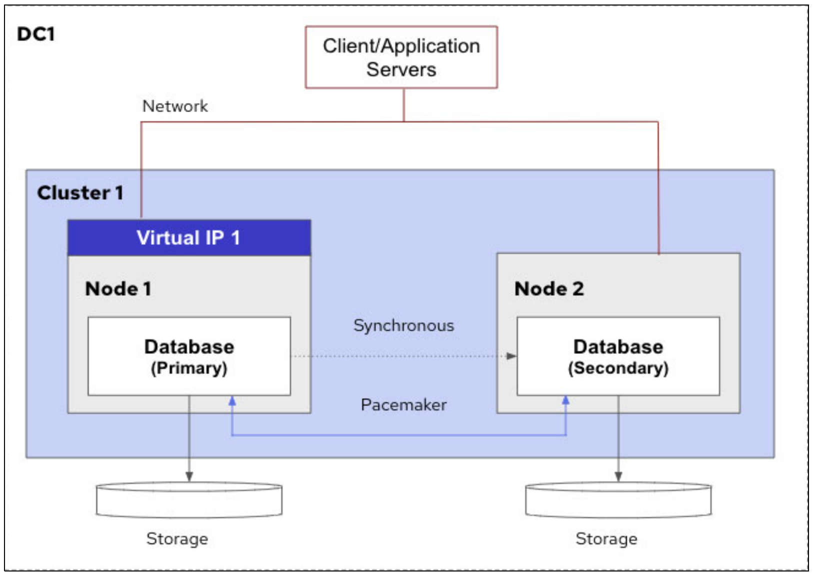


*Figure 2-2 Example of single site Pacemaker cluster architecture for a database with replication*

Figure 2-3 is an example of a single site Pacemaker cluster architecture for general application with shared storage. In this scenario, the application does not have any synchronization method to maintain identical data across the node and can leverage the shared storage configuration in the Pacemaker cluster.

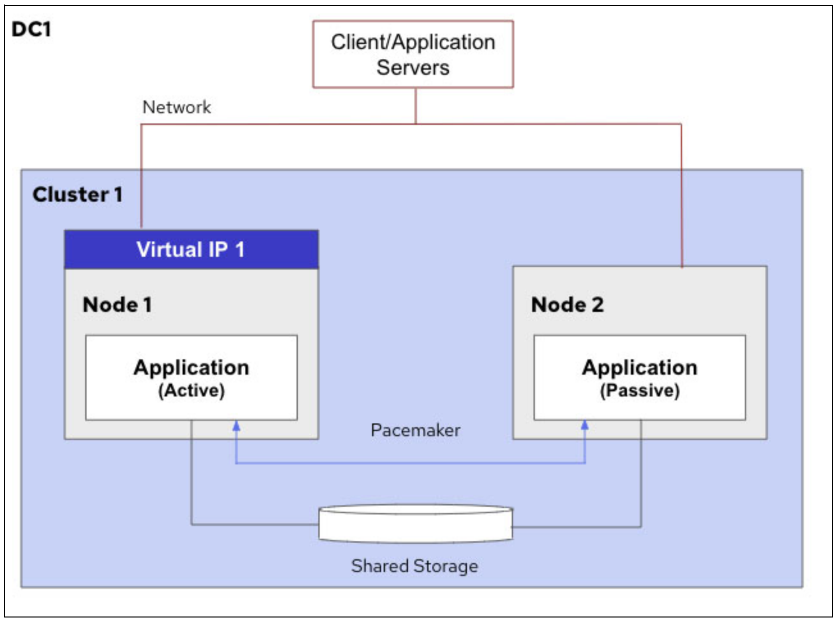


*Figure 2-3 Example of single site Pacemaker cluster architecture for general application with shared storage*

### 2.2.2 Two site cluster

The two-site cluster configuration consists of two clusters[2]: one active and one for DR. A method for creating a two site cluster with DR for an HA cluster is to configure two clusters, one cluster as the primary site cluster and the second cluster as the DR cluster. Alternatively, you could create a single cluster that is managing (stretching) cluster resources across two sites.

For example, a database running in the primary cluster in promoted mode and running in the DR cluster in demoted mode. The database is setup and configured in a way, so that data is synchronized from the primary site to the DR site.

When the primary cluster goes down, the Pacemaker command interface can be used to manually fail the resources over to the DR site. Once the primary cluster has recovered, the Pacemaker command interface can be used to manually move resources back to the primary site.

Figure 2-4 is an example of a two site Pacemaker cluster architecture for a DR scenario.

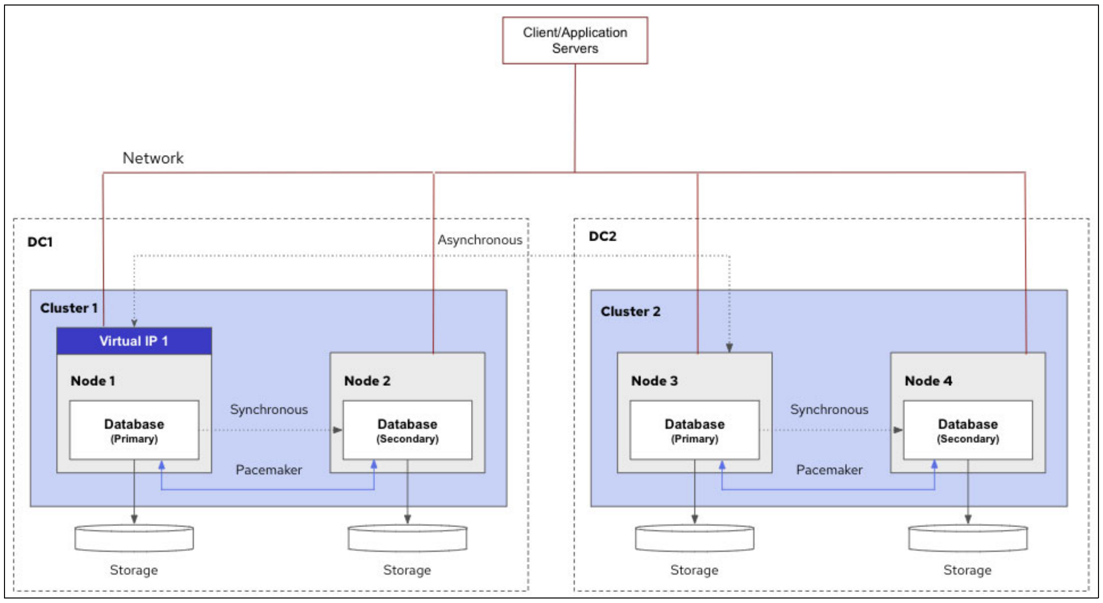


*Figure 2-4 Example of two-site Pacemaker cluster architecture for a DR scenario*

Figure 2-5 on page 33 shows a stretch cluster that is designed to withstand the loss or failure of all members at a given physical site. A stretch cluster comprises a single infrastructure and membership spanning all sites. Membership of the cluster is logically divided into two groups so that cluster services can continue with minimal disruption when an entire group fails or becomes unreachable.

If there is shared storage, it is replicated through hardware or software replication mechanisms so that each group has access to a replica. The groups are typically, but not necessarily, at different physical locations, often with reduced communication inter-connectivity and increased delay compared to a single site. For more information, see Support Policies for RHEL High Availability Clusters - Deployments Spanning Multiple Sites.

[2] `Configuring disaster recovery clusters`

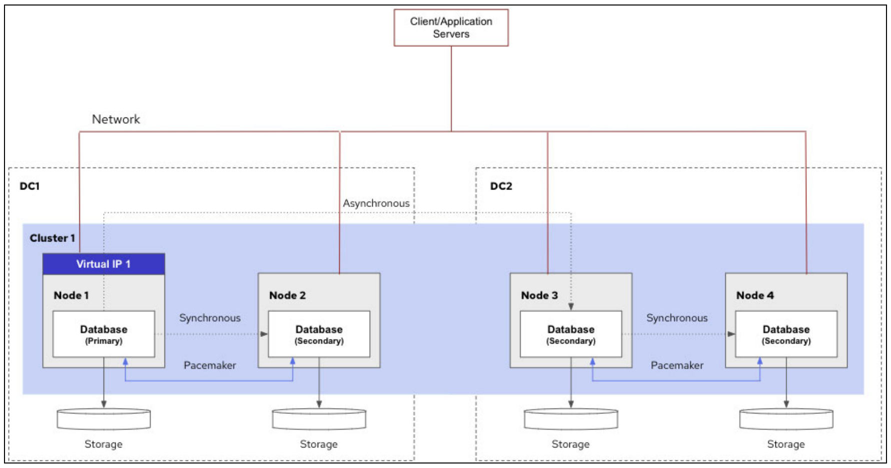


*Figure 2-5 Example of two-site Pacemaker cluster architecture for a stretch cluster*

## 2.2.3 Multisite cluster

Multisite or also known as geo clusters[3], are clusters stretched out over multiple physical locations. Multi-site or disaster-tolerant clusters typically use SAN-based storage replication to replicate data are and usually used in an active/passive manner for DR with manual failover of the active cluster to the passive cluster. This is shown in Figure 2-6.

In a two-site cluster, there will be potential issues with network connectivity between the sites that can lead to split-brain situations. When connectivity drops, there is no way for a node on one site to determine whether a node on another site has failed or is still functioning with a failed site interlink. In addition, it can be problematic to provide high availability services across two sites which are too far apart to keep synchronous. To address these issues, Pacemaker provides full support for the ability to configure high availability clusters that span multiple sites through the use of a Booth cluster ticket manager.

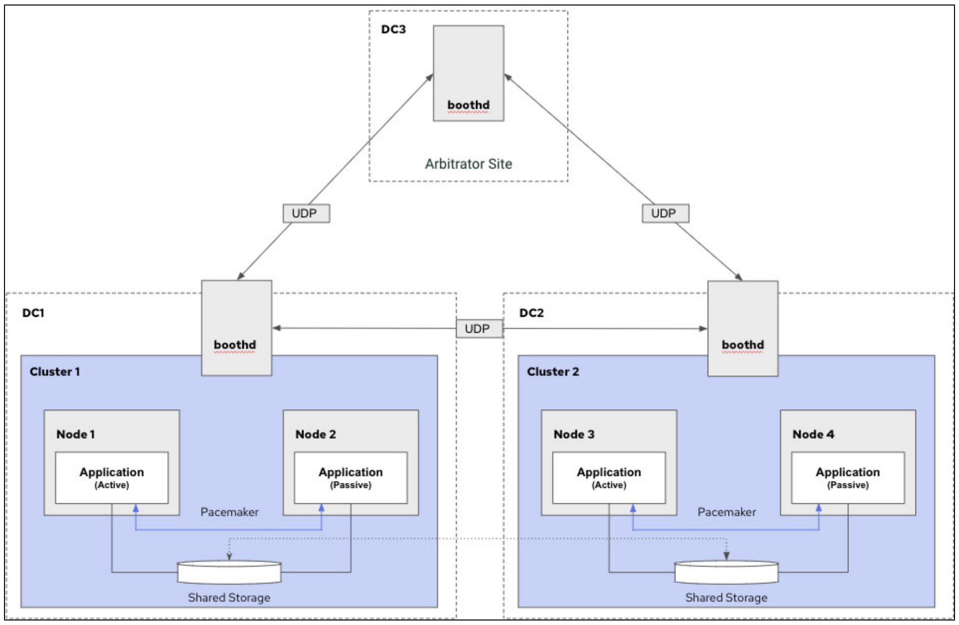


*Figure 2-6 Example of a Multisite cluster with two sites and a single arbitrator on a third site.*

[3] SUSE Linux Enterprise High Availability 15 SP1 (unsupported)

Multisite clusters can be considered overlay clusters where each cluster site corresponds to a cluster node in a traditional cluster. The overlay cluster is managed by the booth cluster ticket manager (in the following called booth).

Each of the parties involved in a geo cluster runs a service, the boothd. It connects to the booth daemons running at the other sites and exchanges connectivity details. For making cluster resources highly available across sites, booths rely on cluster objects called tickets. A ticket grants the right to run certain resources on a specific cluster site. Booth guarantees that every ticket is granted to no more than one site at a time.

If the communication between two booth instances breaks down, it might be because of a network breakdown between the cluster sites or because of an outage of one cluster site. In this case, you need an additional instance (a third cluster site or an arbitrator) to reach consensus about decisions (such as failover of resources across sites). Arbitrators are single machines (outside of the clusters) that run a booth instance in a special mode. Each geo cluster can have one or multiple arbitrators. The most common scenario probably is a multisite or geo cluster with two sites and a single arbitrator on a third site. This requires three booth instances.

## 2.3 Failover policies

Pacemaker offers several failover policies and constraints to manage resources within a cluster environment. These policies and constraints control how resources behave in the event of failures or changes in the cluster. Table 2-1 provides descriptions and examples of some Pacemaker failover policies.

*Table 2-1 Common Pacemaker failover policies*

| Failover Policy | Description | Example |
|---|---|---|
| Priority-Based Failover | Resources are assigned priority levels and Pacemaker attempts to start resources on nodes with the highest priority that are available and suitable for hosting the resources. | In a scenario with two nodes, Node A has a higher priority (for example, 100) for running a critical service compared to Node B (for example, 50). If Node A fails, Pacemaker will try to start the service on Node B. |
| Location Constraints | Defines preferred or forbidden nodes for specific resources, guiding Pacemaker on where to place or avoid relocating resources during failover. | An administrator sets a location constraint that the database service should run on Node X unless Node X fails. In case of a failure on Node X, Pacemaker relocates the service to Node Y, the next preferred node. |
| Collocation Constraints | Groups resources together or enforces separation to specify which resources can or cannot run simultaneously on the same node. | To ensure optimal performance, a web server and a database are collocated, meaning they run on the same node. However, a database and a certain critical service are anti-collocated to avoid resource contention. |
| Ordering Constraints | Defines the sequence in which resources should start or stop to maintain dependencies. | A scenario where a web server must start after a network interface is up. An ordering constraint ensures that the network interface starts before the web server starts. |
| Resource Stickiness | Determines how likely a resource is to stay on its current node unless necessary (for example, node failure or resource failure). | A service has a high stickiness set, so even if a better node is available, Pacemaker avoids moving the service unless the current node fails. |

These examples illustrate how administrators can use Pacemaker failover policies and constraints to ensure resource availability, optimal placement, and controlled failover behavior within a cluster environment. Each policy serves specific needs related to resource management, ensuring reliability and performance in a clustered setup.

The selection of Pacemaker failover policies depends heavily on the specific requirements and characteristics of your clustered environment, as well as the criticality of the services being managed. However, here are some general recommendations for utilizing Pacemaker failover policies effectively:

- Priority-Based Failover: Assign appropriate priority levels to critical resources. Higher priority resources will be allocated on nodes with better capabilities or higher availability.
- Location Constraints: Define preferred nodes for critical services or those with specific hardware or configurations. This ensures services are allocated to nodes best suited for their requirements.
- Collocation Constraints: Use collocation and anti-collocation constraints to control resource placement. Group resources that benefit from running together and avoid conflicts between others.
- Ordering Constraints: Establish dependencies between resources to maintain proper sequencing during start-up or shut-down procedures.
- Resource Stickiness: Adjust stickiness levels carefully. Higher stickiness prevents unnecessary resource movements but may hinder load balancing if set too high.

Remember, there is no one-size-fits-all approach. It is crucial to tailor these failover policies to your specific environment, considering the nature of applications, available resources, and business continuity requirements. Regular monitoring and adjustments based on real-time performance are also essential for maintaining an effective and reliable HA setup using Pacemaker.

3

# Planning and Installing Your Pacemaker Environment

IBM Power Systems are renowned for their robust architecture and exceptional performance. When combined with Pacemaker, they offer a compelling solution for achieving high availability (HA). This chapter serves as a foundational guide to understanding the intricate process of installing and configuring Pacemaker on IBM Power Systems. You are given an overview of the key concepts, prerequisites, and initial steps that are required to embark on your journey toward creating a highly available environment.

This chapter contains the following topics:

- 3.1, “Prerequisites and requirements” on page 38.
- 3.2, “Installing and Configuring the Pacemaker environment” on page 40.

**Note:** While Pacemaker is compatible with several supported Linux distributions on IBM Power System and IBM Power Virtual Server, this chapter primarily focuses on Red Hat Enterprise Linux (RHEL) High Availability Add-On.

## 3.1 Prerequisites and requirements

To deploy Pacemaker on Linux on IBM Power, you need to consider both hardware and software prerequisites:

- Hardware Prerequisites:
  - IBM Power Server and IBM Power Virtual Server: Ensure you have an IBM Power System server with sufficient computing resources (CPU, RAM, and storage) to support your intended workload. The specific model and configuration will depend on your workload requirements.
  - Network Infrastructure: A well-configured network infrastructure is essential for cluster communication. You need at least two network interfaces for redundancy. IBM Power servers typically have built-in network interfaces, but you can add additional network adapters if needed. For more information, see Support Policies for RHEL High Availability Clusters - Cluster Interconnect Network Interfaces.
  - Shared Storage: For HA clustering, shared storage is crucial. You can use technologies like IBM Storage Solutions (for example, IBM Spectrum Scale or IBM FlashSystem) or SAN (Storage Area Network) devices to provide shared storage for your cluster. For more information, see Support Policies for RHEL Resilient Storage.
- Software Prerequisites:
  - Linux Distribution: Choose a Linux distribution that is certified and supported for use with Pacemaker on IBM Power Systems. Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server, and Ubuntu Server are common options:
    - Supported RHEL High Availability releases on IBM Power architectures:

      Red Hat provides support for the following combinations of RHEL HA running on IBM Power architectures. Table 3-1 provides an overview of prerequisites for utilizing RHEL HA components on IBM Power platform.

*Table 3-1 Supported RHEL HA Add-On on Linux on IBM Power*

| IBM Power Server | Supported RHEL HA Add-On version |
|---|---|
| IBM Power8® | Supported with RHEL 8.<br>Supported with RHEL 7.4 or later - using corosync-2.4.0-9.el7 or later and Pacemaker-1.1.16-8.el7 or later. |
| IBM Power9® | Supported with RHEL 8. |
| IBM Power10 | Supported with RHEL 8.4 or later. |

- Support for RHEL High Availability on IBM Power bare-metal servers:

  Table 3-2 describes the support of RHEL HA clusters running on IBM Power bare-metal running OPAL firmware.

*Table 3-2 Supported RHEL HA Add-On on IBM Bare Metal*

| RHEL version | Supported on IBM Bare Metal |
|---|---|
| RHEL 8 | Supported by Red Hat. |
| RHEL 7 | Supported by Red Hat with RHEL 7.5 or later - with corosync-2.4.3-2.el7 or later and Pacemaker-1.1.18-11.el7 or later. |

**Note:** RHEL HA Add-On is not officially supported on IBM PowerKVM and Red Hat does not offer assistance for RHEL HA Add-On in PowerKVM virtualized environments. Additionally, running KVM virtual machines (cluster nodes) on an IBM Power host (hypervisor) is not within the scope of supported configurations.

- IBM Power Virtual Server (PowerVS) LPARs as RHEL HA members:

  Red Hat only supports IBM Power Virtual Server (PowerVS) LPARs as RHEL HA Add-On members on the following versions, shown in Table 3-3.

*Table 3-3 Supported on PowerVS*

| RHEL version | Supported on IBM PowerVS |
|---|---|
| RHEL 8.4.z | Errata RHBA-2023:0503 with the following package(s): fence-agents-ibm-powervs-4.2.1-65.el8_4.11. or later for RHEL 8.4.z. |
| RHEL 8.5.z | Not supported. |
| RHEL 8.6 | RHEL 8.6: Errata RHBA-2023:0428 with the following package(s): fence-agents-ibm-powervs-4.2.1-89.el8_6.8 or later. |
| RHEL 8.7 | Errata RHBA-2023:0843 with the following packages(s): fence-agents-ibm-powervs-4.2.1-103.el8_7.1 or later for RHEL 8.7.z. |

– Pacemaker

  Install the Pacemaker software on all cluster nodes. You can typically find Pacemaker in the distribution's package repositories. Use the package manager (for example, yum for RHEL/CentOS or zypper for SUSE Linux Enterprise Server) to install it.

– Corosync

  Pacemaker relies on Corosync for cluster communication. Ensure Corosync is installed and configured correctly on all cluster nodes.

– Resource Agents

  Depending on your applications and services, you may need resource agents specific to your workload. These agents allow Pacemaker to manage resources like IP addresses, services, and file systems. Install and configure the necessary resource agents.

– Cluster Configuration

  Create a cluster configuration file (typically `/etc/corosync/corosync.conf`) to define cluster properties such as nodes, rings, and quorum settings. Also, configure Pacemaker's cluster configuration (usually in `/etc/cluster/cluster.conf`).

– Cluster Authentication

  Implement secure authentication between cluster nodes. This often involves setting up shared keys or certificates to ensure secure communication.

– Fencing Devices

  Configure fencing devices to ensure the reliability of your cluster. IBM Power Systems may support various fencing methods. The following are requirements around fencing, fence-devices, and STONITH in  a RHEL HA Add-On cluster:

  - STONITH/fencing must be enabled.
  - Every node must be managed by a fence device.
  - Clusters with shared block storage or Distributed Lock Manager (DLM) require power or storage-based devices.

- Clusters with no shared block storage or DLM may use alternative agents and manual fencing.

**Note:** Red Hat support for environments that employ fence agents not officially supplied or endorsed by Red Hat is restricted. In cases where fencing actions are required, Red Hat Support may refrain from providing assistance or engaging in investigations for cluster deployments utilizing non-Red Hat supported fence agents.

– Monitoring and Management Tools

Consider using monitoring and management tools like Hawk or pcs (Pacemaker Configuration System) for easier cluster management and monitoring.

## 3.2 Installing and Configuring the Pacemaker environment

The following are required steps to install and configure Pacemaker on IBM Power Server and IBM Power Virtual Server

1. Install RHEL HA Add-On software
   a. Check that the RHEL HA repository is enabled. On both nodes, use the command shown in Example 3-1.

*Example 3-1 Check RHEL repositories*

```
$ dnf repolist
```

   b. If the RHEL HA repository is missing, use the commands in Example 3-2 to enable it.

*Example 3-2 Enable RHEL HA repositories*

```
$ subscription-manager repos --enable="rhel-8-for-ppc64le-highavailability-e4s-rpms"
$ dnf clean all
$ dnf repolist
```

   c. On both nodes, install the RHEL HA Add-On software packages.
      i. For on-premises IBM Power systems, run the command shown in Example 3-3.

*Example 3-3 Install RHEL HA Add-On on IBM Power System*

```
$ dnf install -y pcs pacemaker fence-agents-all
```

      ii. If you are running on IBM PowerVS instances, use the commands in Example 3-4.

*Example 3-4 Install RHEL HA Add-On on IBM Power Virtual Server*

```
$ dnf install -y pcs pacemaker fence-agents-ibm-powervs
```

   d. Make sure that you install the minimal version of the fence-agents-ibm-powervs package dependent on your Red Hat Enterprise Linux release:
      - RHEL 8.4 fence-agents-ibm-powervs-4.2.1-65.el8_4.11.noarch.rpm
      - RHEL 8.6 fence-agents-ibm-powervs-4.2.1-89.el8_6.8.noarch.rpm

2. Configure Cluster Nodes

   Edit the Corosync configuration file `/etc/corosync/corosync.conf` to define the cluster properties, such as the ring settings and node information. See Example 3-5.

*Example 3-5 Edit Corosync config file*

```
$ sudo nano /etc/corosync/corosync.conf
```

Example 3-6 is an example Corosync configuration.

*Example 3-6 Example of Corosync config file*

```
$ sudo nano /etc/corosync/corosync.conf
totem {
version: 2
secauth: off
cluster_name: mycluster
transport: udpu }
nodelist {
node {
ring0_addr: <Node1_IP>
name: <Node1_Name>
}
node
{ ring0_addr: <Node2_IP>
name: <Node2_Name>
}
}
quorum {
provider: corosync_votequorum
}
```

3. Configure Pacemaker Cluster

   Create a Pacemaker cluster configuration. Edit `/etc/cluster/cluster.conf` as shown in Example 3-7.

*Example 3-7 Edit cluster.conf config file*

```
$ sudo nano /etc/cluster/cluster.conf
```

   In Example 3-8 we provide an example of Pacemaker cluster configuration.

*Example 3-8 Example of cluster.conf file*

```
<?xml version="1.0"?>
<cluster>
<cman expected_votes="2" transport="udpu">
<interface name="eth0" ringnumber="0" mcastaddr="226.94.1.1" mcastport="5405"/> </cman>
<clusternodes>
<clusternode name="<Node1_Name>" nodeid="1">
<fence>
<method name="fence_xvm">
<device name="<Fence_Device_Name>"/>
</method>
</fence>
</clusternode>
<clusternode name="<Node2_Name>" nodeid="2"> <fence> <method name="fence_xvm"> <device
name="<Fence_Device_Name>"/>
```

```
</method>
</fence>
</clusternode>
</clusternodes>
<fencedevices>
<fencedevice name="<Fence_Device_Name>" agent="<Fence_Agent_Name>"/> </fencedevices>
<cfs/>
</cluster>
```

4. Start and Enable Services

   Start and enable the Pacemaker and Corosync services on both cluster nodes with the commands shown in Example 3-9.

*Example 3-9 Start and enable services*

```
$ sudo systemctl start corosync
$ sudo systemctl enable corosync
$ sudo systemctl start pacemaker
$ sudo systemctl enable pacemaker
```

5. Configure Resource Agents

   Depending on your workload, configure Pacemaker resource agents for services like IP addresses, services, and file systems. Example 3-10 shows an example of configuring a simple IP address resource.

*Example 3-10 Configure Resource Agents*

```
$ sudo pcs resource create my_ip ocf:heartbeat:IPaddr2 ip=<Virtual_IP>
cidr_netmask=<Subnet_Mask> op monitor interval=30s
```

6. Create and Start the Cluster

   Create and start the cluster using the **`pcs`** command as shown in Example 3-11.

*Example 3-11 Create and start cluster*

```
$ sudo pcs cluster setup --name mycluster <Node1_Name> <Node2_Name>
$ sudo pcs cluster start --all
```

7. Verify the Cluster Status

   Check the cluster status to ensure it is running correctly. The command is shown in Example 3-12.

*Example 3-12 Verify cluster status*

```
$ sudo pcs cluster status
```

8. Resource Management

   You can manage resources with the **`pcs`** command. For example, to start the IP address resource use the command shown in Example 3-13.

*Example 3-13 Start IP address resource*

```
$ sudo pcs resource enable my_ip
```

4

# HA for Your IBM Power Virtual Server Environment

This chapter provides an introduction to the IBM Power Virtual Server offering. IBM Power Virtual server is a cloud solution based on IBM Power server, which is deployed within IBM data centers across the world, collocated with other IBM Cloud components.

The IBM Power Virtual Server offering is capable of hosting IBM AIX, IBM i, and Linux on IBM Power (Red Hat, SUSE Linux Enterprise Server, and Ubuntu) workloads on IBM Power systems. The IBM Power systems hosting these workloads uses their own dedicated networking infrastructure within the IBM Cloud location. The IBM Power VS offering uses Cisco ACI networking within IBM Cloud facilities to connect with the rest of the IBM Cloud services internally and to connect externally to customer on-premises infrastructure components. The storage connected to IBM Power Virtual Server offering is based on IBM FlashSystem storage subsystems that are directly connected using Fibre channel protocol via dual Brocade SAN fabrics.

This chapter contains the following list of topics and discuss in detail to provide a complete understanding of the IBM Power Virtual Server offering and its building blocks that provide IBM Power Virtual Server as a Service within IBM Cloud including HA options within the IBM Power Virtual Server offering:

- 4.1, “Introduction to the IBM Power Virtual Server offering” on page 44
- 4.2, “IBM Power Virtual Server HA” on page 46
- 4.3, “IBM Power Virtual Server Networking Options” on page 48
- 4.4, “IBM Power Virtual Server storage options” on page 58

## 4.1 Introduction to the IBM Power Virtual Server offering

IBM Power Virtual Server is a hosted infrastructure that is fully virtualized and fully integrated with the IBM Cloud Console and provides public access to consume all services offered in IBM Cloud. The offering provides IBM Power compute-based virtual instances, associated IBM storage and virtual network capabilities that are able to host and run all types of IBM and other operating systems, such as AIX, IBM i and Linux (Red Hat, SUSE Linux Enterprise Server, and Ubuntu) on fully virtualized LPARs. IBM Power Virtual Server is a fully managed Infrastructure as a Service (IaaS) offering based on IBM Power Systems and associated IBM Storage.

IBM Power Virtual Server uses dedicated Cisco ACI networking coupled with virtual networking provided by IBM PowerVM hypervisor within the colocation. IBM Power Virtual Server networking extends to the rest of the IBM Cloud allowing a complete set of networking capabilities to connect with other IBM Cloud services and customer services internally and externally. The network expands to provide secure networking for all types of connectivity to customers, with various configuration options available, and is capable of expanding to connect to all IBM Cloud capabilities in an integrated fashion.

The storage used in IBM Power Virtual Server is based on IBM FlashSystems. There are two types of virtualized storage capabilities available to use for workloads. The Tier 1 storage is capable of up to 10 IOPS/GB and Tier 3 storage is capable of up to 3 IOPS/GB. Customers can choose their tier based on their workload performance requirements.

**Note:** Tier 3 storage may not be suitable for production workloads due to mission-critical application performance requirements.

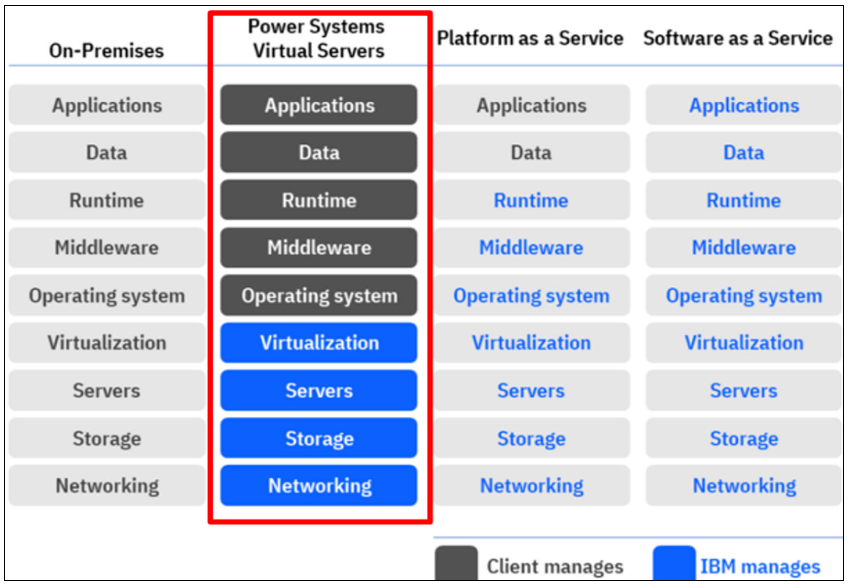
The systems used within IBM Power Virtual Server are IBM Power Systems based on either Power9 or Power10 architecture. The capabilities within these IBM Power systems are no different in this offering from the IBM Power systems that are run by customers on-premise today. They have the same proven IBM advanced Reliability, Availability and Serviceability (RAS) features built in.

The compute used within IBM Power Virtual Server can be scaled up or down. The virtualization layer is fully managed by the IBM Cloud team and is maintained to the highest possible availability based on the deployment model for customer workloads. The location of specific workloads can be designated using affinity or anti-affinity rules to meet production requirements. Additionally, workloads can be pinned to a specific server within, if required for performance or compliance.

The infrastructure is continuously patched and kept up to date by the IBM Cloud operations team. IBM is responsible for managing the hardware to remove any complexities. IBM is responsible for managing the hardware and virtualization layer, while the customer is responsible for managing the operating systems and above. IBM is also responsible for maintaining all IBM storage, Storage Area Network (SAN), and networking hardware, (including firmware) which is patched and kept up to date by the IBM Cloud operations team.

One of the main advantages of using IBM Power Virtual Server is that any enterprise software and workloads that are currently operating on customer premises are certified to run on IBM Power Virtual Server with a simple lift and shift of those workloads. In this method, the workload will continue to run similarly to how it ran previously on-premises. Unlike in on-premises environments, where the customer is expected to maintain currency on hardware, virtualization on IBM Power systems and Storage, the IBM Cloud operations team will maintain and provide timely updates and upgrades with no disruption to running workloads allowing customers to focus more on actual business workloads rather than the infrastructure from operating systems and above on each LPAR.

Figure 4-1 provides a responsibility matrix showing the services management roles and responsibilities by either the IBM Cloud team or client responsibilities for IBM Power Virtual Server.



| On-Premises | Power Systems Virtual Servers | Platform as a Service | Software as a Service |
|---|---|---|---|
| Applications | Applications | Applications | Applications |
| Data | Data | Data | Data |
| Runtime | Runtime | Runtime | Runtime |
| Middleware | Middleware | Middleware | Middleware |
| Operating system | Operating system | Operating system | Operating system |
| Virtualization | Virtualization | Virtualization | Virtualization |
| Servers | Servers | Servers | Servers |
| Storage | Storage | Storage | Storage |
| Networking | Networking | Networking | Networking |

⬛ Client manages 🟦 IBM manages

*Figure 4-1 IBM Power Systems Virtual Server responsibility assignment matrix*[1]

## 4.1.1 IBM Power Virtual Server Compute Hardware Building Blocks

IBM Power Virtual Server deployments are built using select hardware models based on the following IBM Power Hardware systems:

– IBM Power9 Enterprise E980 Systems (9080-M9S)
– IBM Power9 Scaleout S922 Systems (9009-22A or 9009-22G)
– IBM Power10 Enterprise E1080 Systems (9080-HEX)
– IBM Power10 Scaleout S1022 Systems (9105-22A)

For more information, see IBM Power Virtual Server High availability and disaster recovery.

**Note:** The availability of IBM Power systems may vary from one IBM Cloud location to another. Additional IBM Power systems may get deployed over time to each location.

## 4.1.2 IBM Power Virtual Server Network Hardware Building Blocks

IBM Power Virtual Server deployments use the following network infrastructure devices and capabilities as building blocks to provide all network services within IBM Power Virtual Server Offering:

– Cisco Nexus 9000 (25Gb[2])
– Cisco Nexus 9000 (10G)
– Cisco Nexus 9000 (1G)
– Cisco ASR1001-HX Router
– Cisco UCS - APIC controller

[1] What is IBM Power Virtual Server?
[2] Latest Deployments on S1022 and E1080 systems will be using 100Gb capable SR-IOV adapters.

– Avocent ACS 8048, ACS 8016, ACS 8032 DAC-400

**Note:** The deployment of network devices may vary from one IBM Cloud location to another. Additional devices and technologies may get deployed over time.

### 4.1.3 IBM Power Virtual Server Storage Hardware Building Blocks

IBM Power Virtual Server deployments are using 32GB Brocade SAN infrastructure with following types of devices as building blocks as storage providers.

– IBM FlashSystems Storage based on IBM FS9x00 series devices
– IBM Storwize V7000 with SSD drives
– Brocade 32Gb SAN

**Note:** The deployment of IBM Storage devices may vary from one IBM Cloud location to another. Additional devices and technologies may get deployed over time.

## 4.2 IBM Power Virtual Server HA

This section provides an overview of IBM Power Systems Virtual Server infrastructure-related HA capabilities and DR options available within IBM Cloud deployments.

The IBM Power Virtual Server offering is designed with the concept of eliminating any single point of failure (SPOF) within the underlying infrastructure deployed to host the IBM AIX, IBM i, or Linux on Power (Red Hat, SUSE, and Ubuntu) workloads. This is achieved through the use of redundant physical hardware components and virtualization technologies built using IBM Power Systems availability features.

The use of dual hardware adapters, each connected to different storage connections or network switches, provides basic HA and eliminates any effect of a failure of any single hardware component. The IBM Power Virtual Server infrastructure also uses dual SAN Fabrics for storage access, providing availability in case one fabric fails as the secondary fabrics will continue to service the workloads without any outages. Similarly, there are multiple network switches used to eliminate any SPOF for network gear failures utilized by the IBM Power Virtual Server offering.

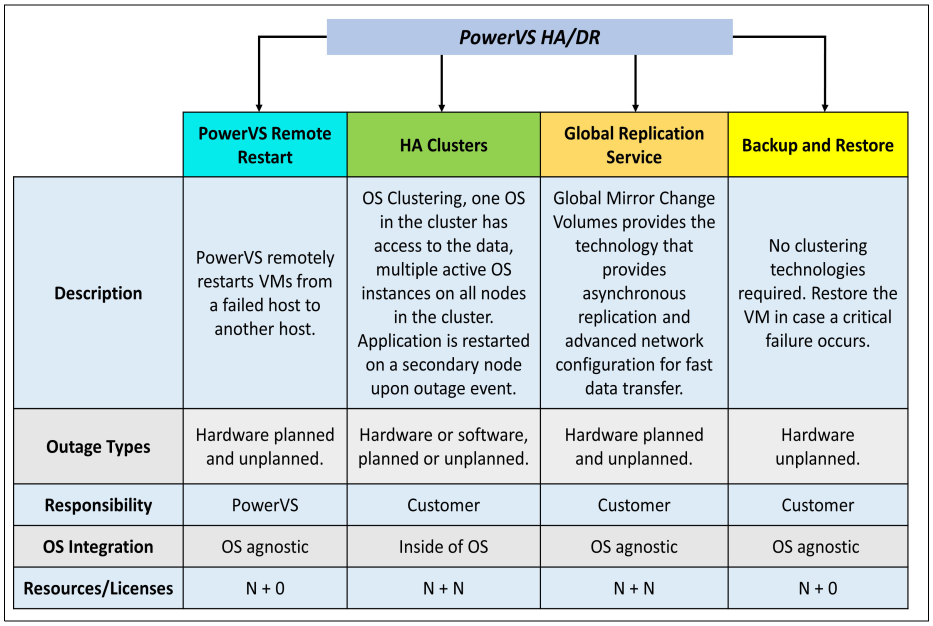
The Network and Storage IO HA within each system is achieved using PowerVM dual Virtual IO Servers that work in pairs within each IBM Power system building block. This allows each system to service workloads with no outages during service maintenance. If any Virtual I/O (VIO) Server fails, all services continue to operate and be available through the remaining VIO Server. In the unlikely case of dual failures or entire IBM Power system failure, the workloads are able to be remotely restarted on another server allowing the workloads to resume services automatically using the remote restart capability built into the IBM Power Virtual Server offering. This level of workload remote restart capability provides basic DR capabilities to continue your business critical applications and ensures that services are automatically relocated and restarted within IBM Power Systems Virtual Server offering.

For critical workloads requiring more sophisticated HA solutions, consider deploying them on IBM Power Virtual Server within a single site across various availability zones or between different regions on IBM Cloud using geographic or metropolitan mirroring techniques.
IBM Power Virtual Servers supports the deployment of IBM PowerHA SystemMirror providing

automatic application failovers for workloads in your IBM Power Virtual Server environment running on AIX or IBM i.

For Linux workloads, Pacemaker cluster solutions can be deployed with advanced application and middleware configurations to automatically monitor and failover workloads based on your application HA/DR requirements.

When applications or databases are deployed to run on IBM Power Virtual Server, there are many options available to maintain HA and business continuity for such workloads. Figure 4-2 describes different business continuity plans that can be used with the IBM Power Virtual Server Offering.


PowerVS HA/DR

| | PowerVS Remote Restart | HA Clusters | Global Replication Service | Backup and Restore |
|---|---|---|---|---|
| **Description** | PowerVS remotely restarts VMs from a failed host to another host. | OS Clustering, one OS in the cluster has access to the data, multiple active OS instances on all nodes in the cluster. Application is restarted on a secondary node upon outage event. | Global Mirror Change Volumes provides the technology that provides asynchronous replication and advanced network configuration for fast data transfer. | No clustering technologies required. Restore the VM in case a critical failure occurs. |
| **Outage Types** | Hardware planned and unplanned. | Hardware or software, planned or unplanned. | Hardware planned and unplanned. | Hardware unplanned. |
| **Responsibility** | PowerVS | Customer | Customer | Customer |
| **OS Integration** | OS agnostic | Inside of OS | OS agnostic | OS agnostic |
| **Resources/Licenses** | N + 0 | N + N | N + N | N + 0 |

*Figure 4-2 IBM Power Virtual Server HA and DR Business Continuity Plan*

The HA solutions can be created in the area of HA clusters in Figure 4-2 for many applications and services running on Linux operating systems on the IBM Power VS solution.

One of the major limitations when deploying HA clusters with Pacemaker or PowerHA within IBM Power VS offering is that there is no way to stop an active node like you would in an on-premise environment where you have access to the Hardware Management Consoles (HMCs), Novalink, or VIO Servers within your infrastructure. To manage and maintain a cohesive cluster environment without the risk of a "split-brain" situation, it is essential to have the ability to halt or fence individual nodes. This action allows you to isolate problematic nodes and prevent them from interfering with the rest of the system, ensuring continued operation and avoiding potential inconsistencies. By fencing a node, you effectively take it offline, isolating its resources and preventing further communication between nodes until the issue has been resolved. IBM PowerVS supports a fence agent (fence_ibm_powervs), which is the only supported agent for a STONITH device on Power Virtual Server clusters. The fence agent connects to the Power Cloud API using parameters to identify an IBM Power Virtual Server instance to stop the resource as required.

## 4.3 IBM Power Virtual Server Networking Options

IBM Power Virtual Server networking is built using multiple virtualization and physical technologies to securely communicate and operate with multiple tenants in a secure fashion maintaining their segregation and isolation as a priority at all levels of networking.

When an IBM Power Virtual Server instance is created, it can be placed on two types of networks at an OS level.

- **Public Network:**

  The public network is an easy and fast method to connect over the internet and stay protected by a firewall allowing few connections through to the OS instance:

  - SSH
  - 5250 Terminal emulation with SSL
  - HTTPS
  - Ping

  This network connection is used for initial configuration over a public network to an OS Instance where there is no private connectivity available. It is common to remove this interface after the initial deployment of a Virtual Server instance and then only use private network interfaces.

- **Private Network:**

  The private network allows to have an interface separate from the public interface, which is allowed to communicate with the rest of the OS instances within the same private VLAN once enabled.

  This network can be used to communicate with various IBM Cloud services and application services as needed by the customer. This can be extended to communicate with external networks in a secure fashion as required.

**Note:** The public network interface can be removed from the IBM Cloud Console at any time and added if needed.

### 4.3.1 IBM Power Systems Virtualization

IBM Power Systems are built with a rich set of virtualization features that are capable of running enterprise class workloads in a secure way, utilizing the IBM PowerVM Hypervisor that enables these virtualized networking features and capabilities on IBM Power hardware.

Some of the relevant key features that are used within PowerVM virtualization technology are discussed in this section.

- **Network Virtualization:**

  IBM PowerVM is implemented with IEEE VLAN-compatible virtual switches in the hypervisor that are capable of providing up to 4,094 VLANs per virtual switch. This provides an abstraction layer between physical network hardware and LPARs and allows the sharing of one or more physical resources across multiple LPARs while maintaining the segregation and separation between each VLAN as per industry standards.

  The VLAN concept allows two LPARs to communicate with each other using the channels between them over the hypervisor vSwitch without leaving the hypervisor, which is the default mode of operation. The virtual switches are able to operate in two different modes:

– **Virtual Ethernet Bridge (VEB) Mode:**

In this mode, all LPARs within a chassis on a virtual switch are allowed to communicate with each other freely within their respective VLANs using IP addresses at layer three. However, the VLAN boundary is not allowed to be crossed, maintaining the VLAN separation at layer two. This is the default mode of operation for vSwitches when created in the Power Hypervisor.

Figure 4-3 illustrates VEB mode operations with a vSwitch.

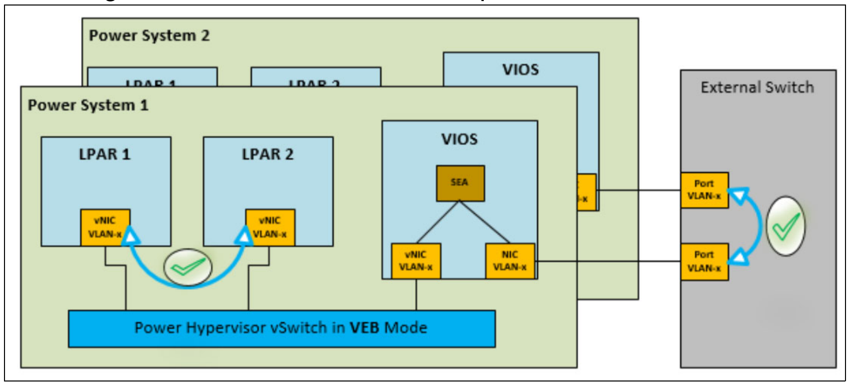


*Figure 4-3 IBM Power Hypervisor vSwitch in VEB mode*

– **Virtual Ethernet Port Aggregator mode:**

In Virtual Ethernet Port Aggregator (VEPA) mode, none of the LPARs within a chassis on a virtual switch are able to communicate with each other, even if they are placed on the same VLAN ID with an IP address. The only way to communicate with another LPAR within the same chassis on the same vSwitch is to send the network packets to an external switch and to use reflective relay capability on external switches to send the packet back into the same vSwitch destined for another LPAR. The reflective relay allows the network traffic to perform a hairpin turn around and the packets return back to the vSwitch where it originated and then delivered to its intended destination at layer three. Without using this reflective relay capability, two LPARs placed on a VEPA mode virtual switch will not be able to communicate successfully. Figure 4-4 illustrates VEPA mode restrictions imposed by the vSwitch.

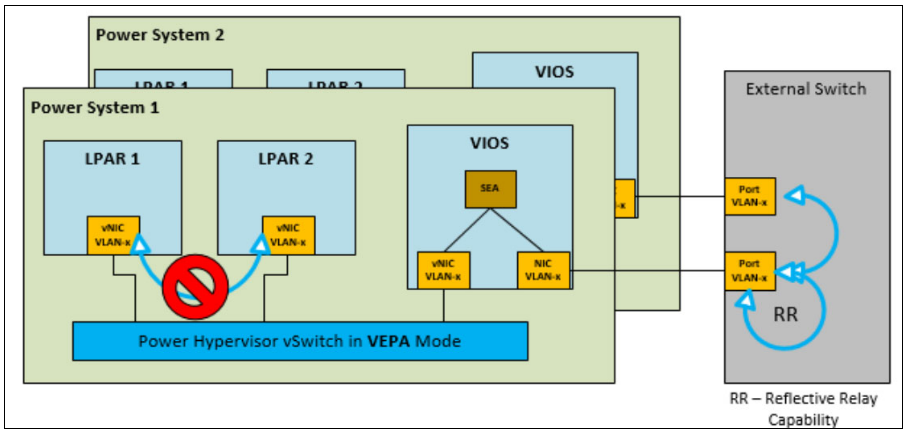


*Figure 4-4 IBM Power Hypervisor vSwitch in VEPA mode*

The network virtualization of physical hardware adapters is achieved by the use of a VIO Server. There are many different options available to implement and connect using a VIO

Server to allow Ethernet bridging or trunking to external networks. Single Root IO Virtualization (SR-IOV) is often used for client LPARs for additional adapter sharing together with the Shared Ethernet Adapter (SEA) within IBM Power Virtual Server deployments in IBM Cloud.

The IBM Power Virtual Server implementation uses two slightly different networking deployments depending on the models of the IBM Power System being utilized.

- **IBM S922 Systems:**

  As S922 Systems are limited in the number of available adapter slots, we use three 2-port adapters that are shared using SR-IOV Virtual Function (VF) between the two VIO Servers that service management, public, and private networking for an S922 System.

  The SEA configuration uses multiple SR-IOV based VFs to aggregate two ports coming from different adapters to eliminate any SPOF on each VIO Server. In addition, this aggregation provides a higher bandwidth to the clients on a system.

  Refer to Figure 4-5 to understand how networking connections are deployed and how they are used for different purposes in the S922 systems.

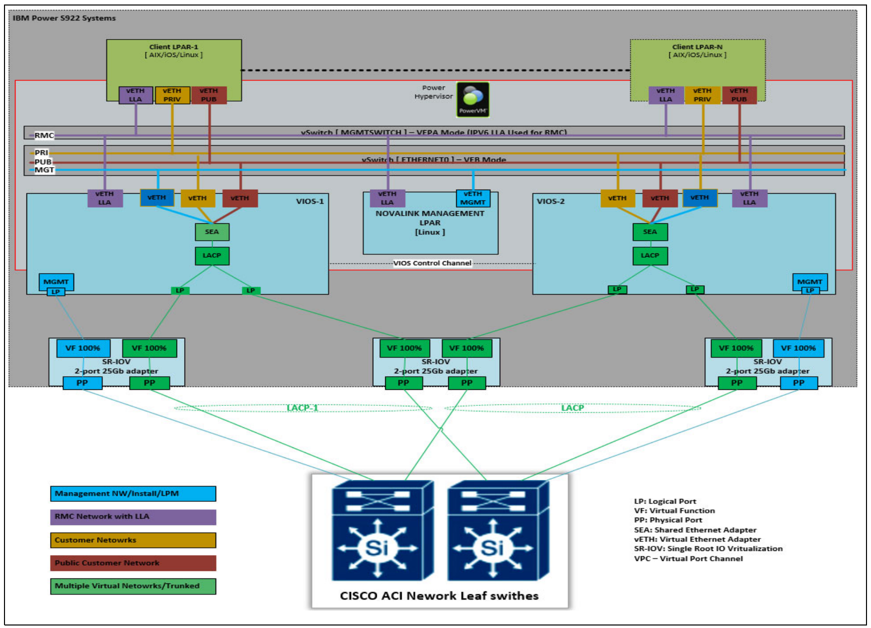

IBM Power S922 Systems
Client LPAR-1
[ AIX/iOS/Linux ]
Client LPAR-N
[ AIX/iOS/Linux ]
vETH LLA
vETH PRIV
vETH PUB
Power Hypervisor
PowerVM
RMC
vSwitch [ MGMTSWITCH ] – VEPA Mode (IPV6 LLA Used for RMC)
PRI
PUB
MGT
vSwitch [ ETHERNET0 ] – VEB Mode
vETH LLA
vETH
VIOS-1
vETH MGMT
NOVALINK MANAGEMENT LPAR [Linux ]
VIOS-2
SEA
LACP
VIOS Control Channel
MGMT
LP
VF 100%
SR-IOV 2-port 25Gb adapter
PP
LACP-1
LACP
Management NW/Install/LPM
RMC Network with LLA
Customer Netowrks
Public Customer Network
Multiple Virtual Netowrks/Trunked
CISCO ACI Nework Leaf swithes
LP: Logical Port
VF: Virtual Function
PP: Physical Port
SEA: Shared Ethernet Adapter
vETH: Virtual Ethernet Adapter
SR-IOV: Single Root IO Vritualization
VPC – Virtual Port Channel

*Figure 4-5 IBM Power VS S922 Network Virtualization*

- **IBM S1022 Systems:**

  The new deployments within IBM Power Virtual Server with IBM Power S1022 systems use 100 Gb SR-IOV adapters and have slightly different architectures than IBM Power S922 systems. The SR-IOV capabilities are not used and the physical ports from SR-IOV adapters are Link Aggregated instead. The same level of HA is achieved by failing over to secondary VIO Servers in case of an adapter failure. This architecture is illustrated in Figure 4-6 on page 51. Power 10 S1022 systems are currently available in MAD02, MAD04, WDC07, and DAL10. Additional S1022 systems will be rolled out to existing and new data centers over time, based on client demand.

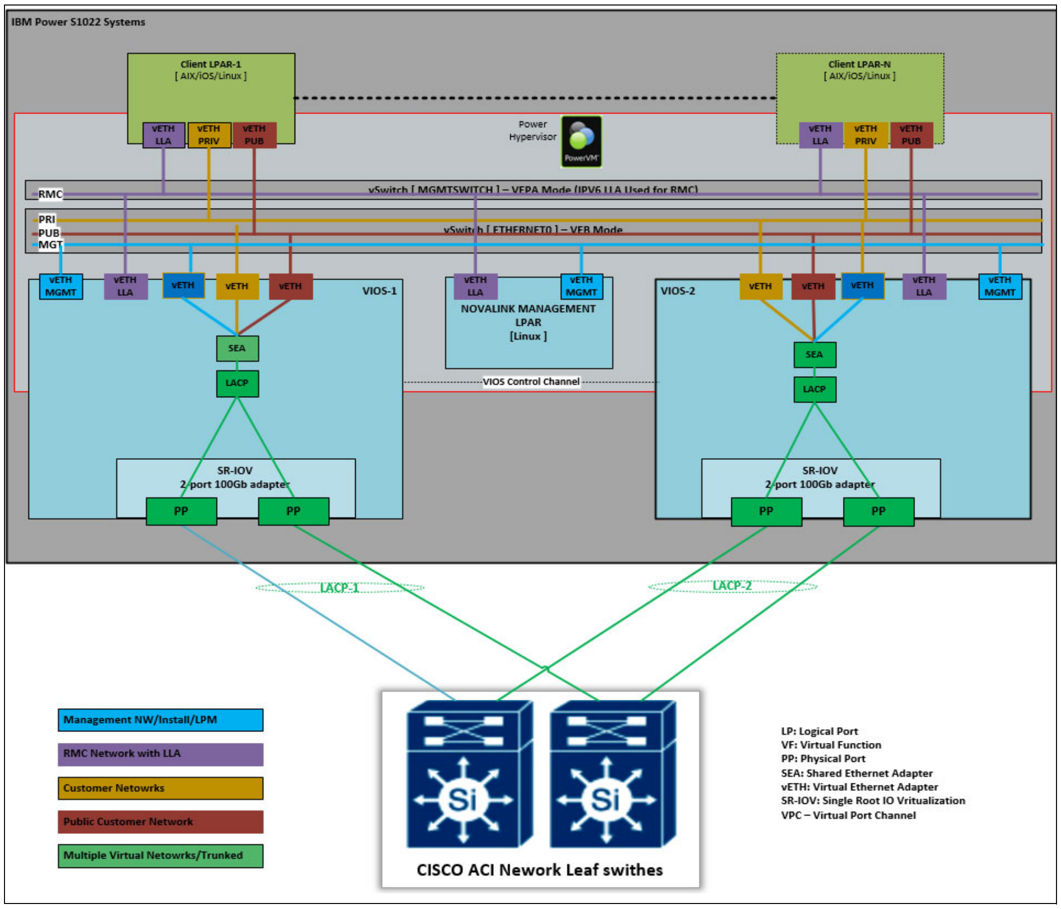

IBM Power S1022 Systems
Client LPAR-1
[ AIX/iOS/Linux ]
Client LPAR-N
[ AIX/iOS/Linux ]
vETH LLA
vETH PRIV
vETH PUB
Power Hypervisor
PowerVM
vSwitch [ MGMTSWITCH ] – VEPA Mode (IPV6 LLA Used for RMC)
RMC
PRI
PUB
MGT
vSwitch [ ETHERNET0 ] – VEB Mode
vETH MGMT
vETH LLA
vETH
VIOS-1
NOVALINK MANAGEMENT LPAR [Linux ]
VIOS-2
SEA
LACP
VIOS Control Channel
SR-IOV 2-port 100Gb adapter
PP
LACP-1
LACP-2
Management NW/Install/LPM
RMC Network with LLA
Customer Netowrks
Public Customer Network
Multiple Virtual Netowrks/Trunked
CISCO ACI Nework Leaf swithes
LP: Logical Port
VF: Virtual Function
PP: Physical Port
SEA: Shared Ethernet Adapter
vETH: Virtual Ethernet Adapter
SR-IOV: Single Root IO Vritualization
VPC – Virtual Port Channel

*Figure 4-6 IBM Power Virtual Server S1022 Network Virtualization*

– **IBM E980 Systems:**

The E980 systems follow a similar pattern, but with more physical adapters as they are capable of having more adapter slots. There are 10 x 2-port SR-IOV adapters in use for each of these systems and they use each port as an SR-IOV VF to create an SEA that services the management, public, and private networks in each frame using dual VIO Servers.

The use of additional adapters/ports increases the network availability and bandwidth per VIO Server when connecting to the external networks. Multiple adapters are aggregated using IEEE LACP using multiple VFs provided by SR-IOV adapter ports to the external Cisco ACI network.

Refer to Figure 4-7 on page 52 to understand how the networking connections are set up and how they are used for different networking solutions with each E980 system.

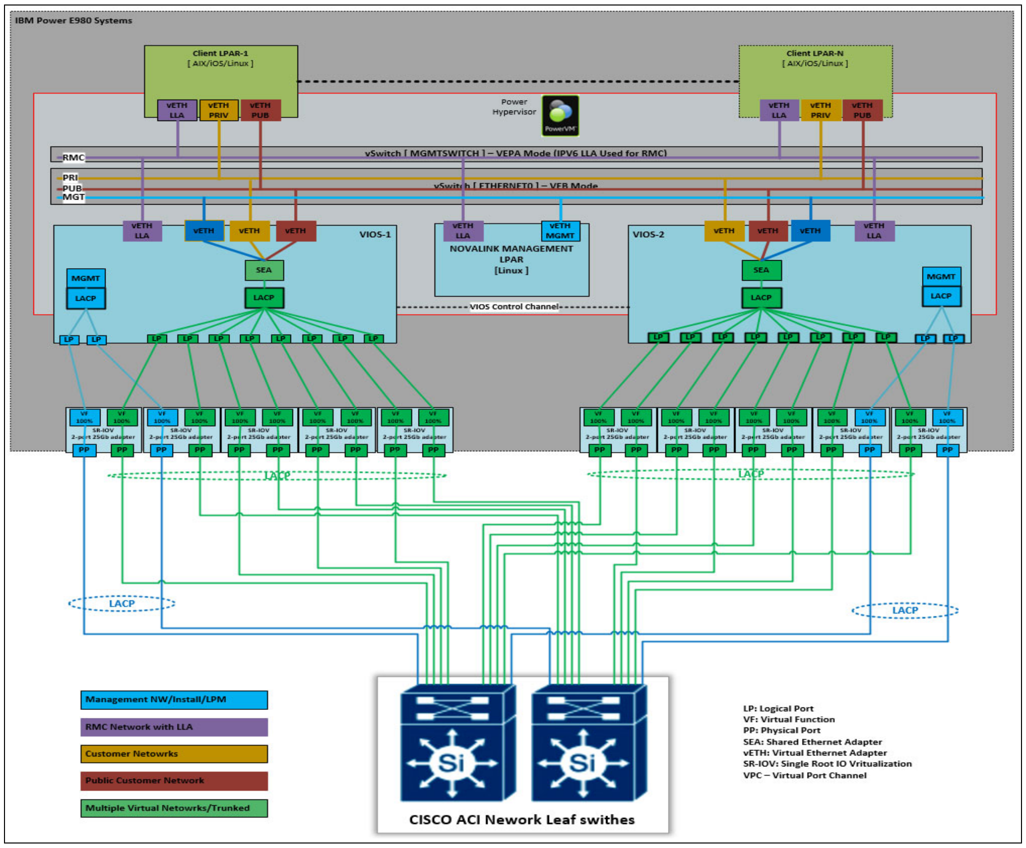

IBM Power E980 Systems
Client LPAR-1 [ AIX/iOS/Linux ]
Client LPAR-N [ AIX/iOS/Linux ]
vETH LLA
vETH PRIV
vETH PUB
Power Hypervisor
PowerVM
RMC
PRI
PUB
MGT
vSwitch [ MGMTSWITCH ] – VEPA Mode (IPV6 LLA Used for RMC)
vSwitch [ ETHERNET0 ] – VEB Mode
vETH
VIOS-1
VIOS-2
vETH MGMT
NOVALINK MANAGEMENT LPAR [Linux ]
SEA
MGMT
LACP
VIOS Control Channel
LP
VF 100%
SR-IOV 2-port 25Gb adapter
PP
LACP
Management NW/Install/LPM
RMC Network with LLA
Customer Netowrks
Public Customer Network
Multiple Virtual Netowrks/Trunked
CISCO ACI Nework Leaf swithes
LP: Logical Port
VF: Virtual Function
PP: Physical Port
SEA: Shared Ethernet Adapter
vETH: Virtual Ethernet Adapter
SR-IOV: Single Root IO Virtualization
VPC – Virtual Port Channel

*Figure 4-7 IBM Power VS E980 Network Virtualization*

– **IBM E1080 Systems:**

The new deployments within IBM Power Virtual Server for E1080 systems are to be used with multiple 100 Gb SR-IOV adapters. The architecture is being currently tested at the time of writing this publication. The most likely configuration will be based on the use of multiple 100Gb physical ports using SR-IOV adapters on a Link Aggregation Control Protocol (LACP). Deployment of E1080 systems will be based on customer demand.

► **Storage Virtualization:**

There are three main methods wherein a client LPAR can map external or internal storage used in IBM Power Systems using VIO Servers.

– Shared Storage Pools [SSP]
– Virtual SCSI [vSCSI]
– N-Port Id Virtualization [NPIV]

Each of above virtualization methods uses a different method to provide virtualized storage to LPARs hosted on an IBM Power System. In IBM Power VS deployments, SSPs are not in use for any purpose. Hence, SSPs are not discussed.

**Virtual SCSI**

The Virtual SCSI (vSCSI) option is used between VIO Servers and client LPARs to map disks that are visible on VIO Servers. Each VIO Server maps the same SAN disk device provisioned from external SAN storage. The use of vSCSI is limited for to the Novalnk LPAR and is not used with any client LPARs. The boot disk for Novalink LPAR is mapped from each VIO Server using vSCSI.

The disk is a protected highly available volume provided by the IBM Storage subsystem eliminating any requirement to set up mirroring on the Novalink partition.

**N-Port ID Virtualization**

N-Port ID Virtualization (NPIV) is an industry standard technology that provides the capability to assign a physical Fibre Channel adapter to multiple unique world wide port names (WWPN). To access physical storage from a SAN, the physical storage is mapped to logical units (LUNs) and the LUNs are mapped to the ports of physical Fibre Channel adapters. Then the PowerVM VIO Server uses the maps to connect the LUNs to the virtual Fibre Channel adapter of the virtual I/O client as a pass-through virtual adapter using PowerVM virtual slots.

IBM PowerVS uses NPIV for all client LPARs to map storage from external storage providers. With the use of dual PowerVM VIO Servers, each mapped LUN is mapped providing highly available volumes to the client LPARs through Power VM virtualization layer maintaining a minimum of two paths from VIO Servers to maintain availability and performance with the use of multipath device drivers at the client Operating system level. The use of multipath device drivers will depend on the type of operating systems.

Refer to Figure 4-8 for further details on how vSCSI and NPIV connectivity is deployed using physical adapters on IBM Power S922/S1022 Systems.

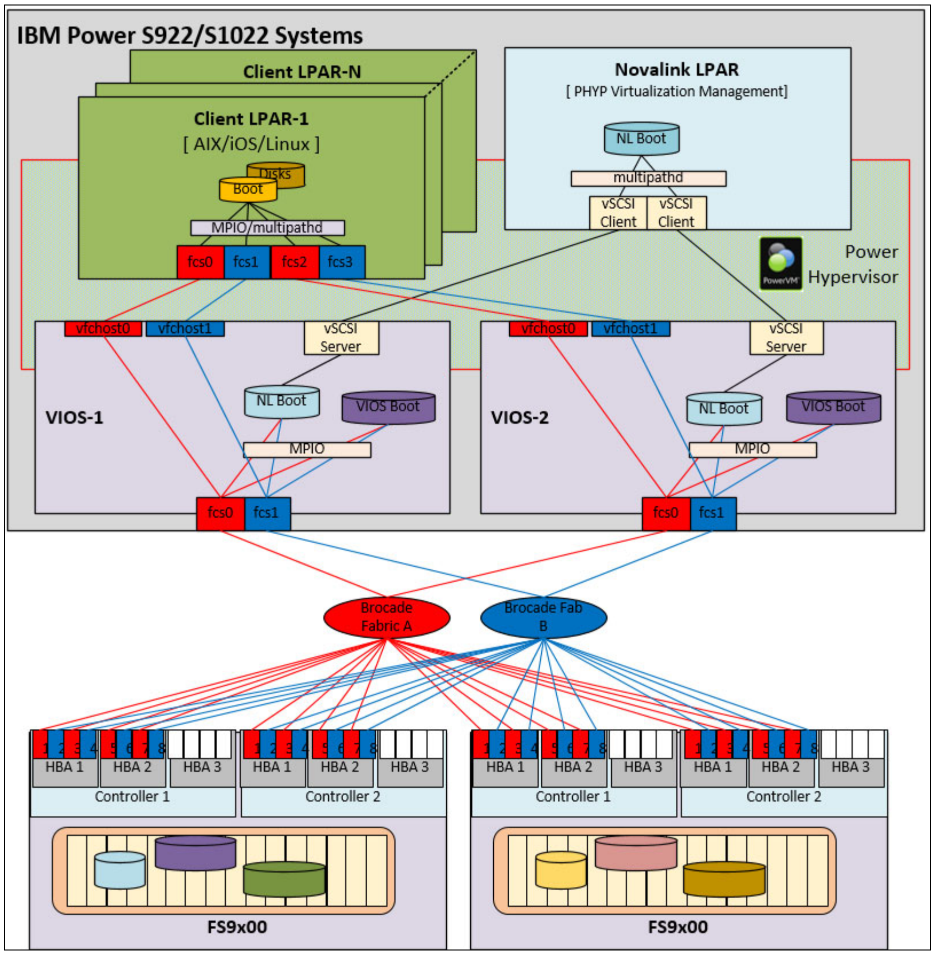


*Figure 4-8 IBM Power Virtual Server vSCSI and NPIV FC Connectivity on IBM Power S922/S1022 Systems*

Refer to Figure 4-9 for further details on how vSCSI and NPIV connectivity is deployed using physical adapters on IBM Power S922/S1022 Systems.

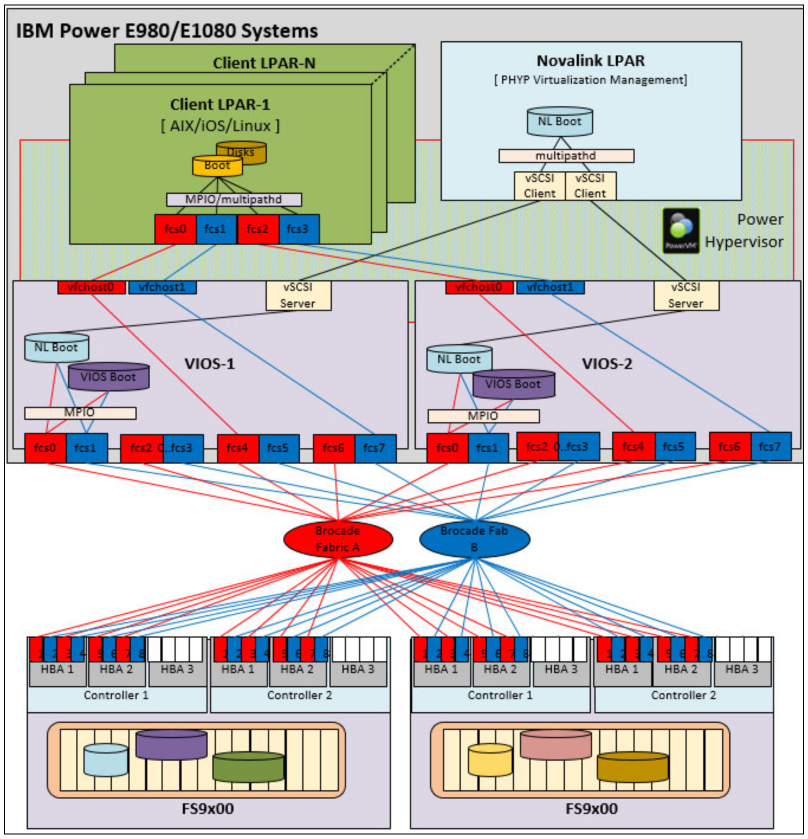


Figure 4-9 IBM Power Virtual Server vSCSI and NPIV FC Connectivity on IBM Power E980/E1080 Systems

- **IBM PowerVM NovaLink:**

  NovaLink is a software interface for virtualization management. The NovaLink is installed on a PowerVM server within a Linux LPAR. PowerVM NovaLink enables highly scalable modern cloud management and deployment of critical enterprise workloads. The PowerVM NovaLink is used to provision large numbers of VMs on PowerVM servers quickly and in a robust way using OpenStack services. You can manage the server through a representational state transfer (REST) application programming interface (API) or through a command-line interface (CLI). NovaLink seemlessly integrates with IBM and non-IBM Cloud managers such as PowerVC or IBM Cloud Console using REST API and Openstack.

  The NovaLink runs on a Linux LPAR on a POWER8, POWER9, or Power10 processor-based systems that is virtualized by PowerVM. Each frame must have this Novalink LPAR deployed for virtualization management using Novalink capabilities.

  IBM PowerVM Novalink distributes the virtualization management to each frame rather than from a single point like a Hardware Management Consoles (HMC). Using Novalink enables a larger number of LPARs and IBM Power systems to be managed compared to utilizing HMC management. This feature allows the maximum scalability to provide more virtual machines managed with better distribution across the manged servers.

PowerVM Novalink is a single instance that is running on each IBM Power System and may fail /restart without impacting any active and running workloads on such frames. Hence, it is not seen as a SPOF. If the Novalink Partition is not operational, there may be limits to the ability to make changes to any LPAR on the frame with dynamic resource changes, console access, live partition mobility actions and any power control on an affected IBM Power system. For more information, see PowerVM NovaLink.

### 4.3.2 Cisco ACI Networking

IBM Power Virtual Server uses Cisco's premier SDN solution, Application Centric Infrastructure (ACI) networking, which enables multicloud environments with agility and resiliency. The IBM Power VS offering uses the Cisco ACI Spine and Leaf switch configuration to provide ACI capability. For more information, see Cisco Application Centric Infrastructure (Cisco ACI) Solution Overview.

### 4.3.3 External Networking

This section describes the available external network connectivity with IBM Power VS. This is not an exhaustive list of connections methods that are available today. The deployment of IBM Power VS makes use of either public or private network interface within an LPAR.

- **Internet:**

  Access the internet through resources that are hosted in any of the following infrastructure environments using layer three IPv4 traffic:

  - IBM Cloud Classic
  - IBM Virtual Private Cloud (VPC)
  - IBM PowerVS

- **Remote:**

  Connect remote networks to your IBM Cloud networks using the following services that allow customers to connect to a remote network in IBM Cloud:

  - **Internet VPN:**

    Access over the public internet to connect remote networks to IBM Cloud networks using a VPN. The VPN is terminated on gateway devices or a service within IBM Cloud.

  - **Megaport or Direct Link 2.0 Connectivity:**

    Direct Link is a suite of offerings that enable the creation of direct, private connections between your remote, on-premises network and IBM Cloud, without traversing the public internet.

    Figure 4-10 on page 56 illustrates Megaport or Direct Link 2.0 connectivity from IBM Power Virtual Server offering to on-premise networks.

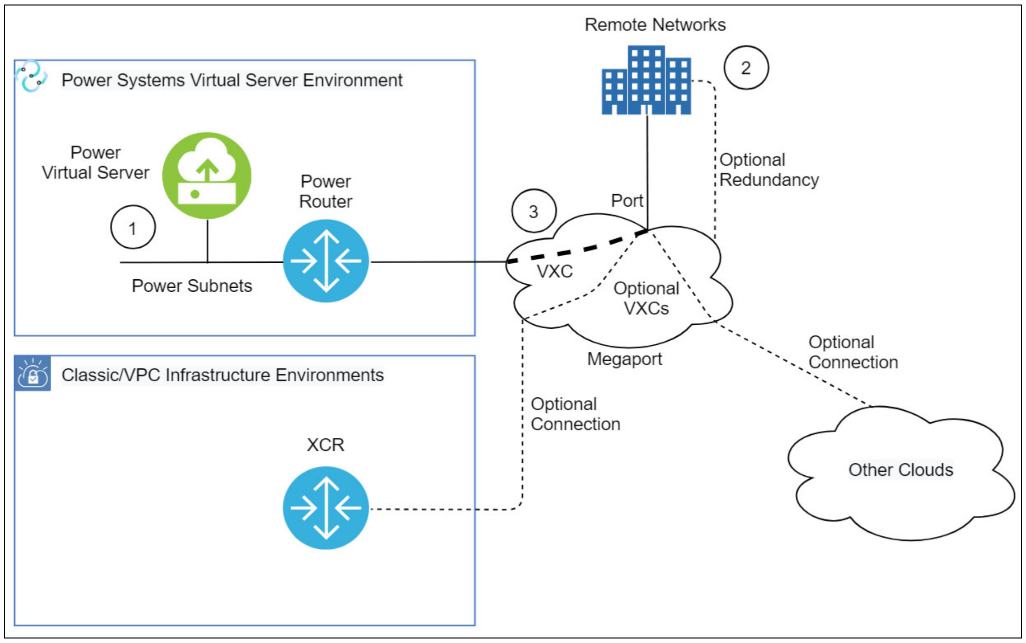


*Figure 4-10 Connect to on-premise networks*

The use of Direct Link 2.0 or Megaport seamlessly connects your on-premises resources to your cloud resources. The speed and reliability of Direct Link extends your organization's data center network and offers more consistent, higher-throughput connectivity, keeping traffic within the IBM Cloud network.

The Direct Links can connect to either a local or remote IBM Cloud Transit Gateway, which allows the on-premises network to access all networks that are connected to the IBM Cloud Transit Gateway.

Your Direct Link 2.0 or Megaport connections are location-specific. By default, IBM Cloud Direct Link is not a redundant service. You must order a separate Direct Link Connect instance for redundancy.

Transit Gateway enables you to connect your otherwise disconnected private networks, such as classic, VPC, and Direct Link. This allows you to establish a connection between multiple IBM Power Systems Virtual Server workspaces across different data centers. Figure 4-11 on page 57 illustrates connectivity between multiple Power Systems Virtual Server locations with HA and DR solutions.

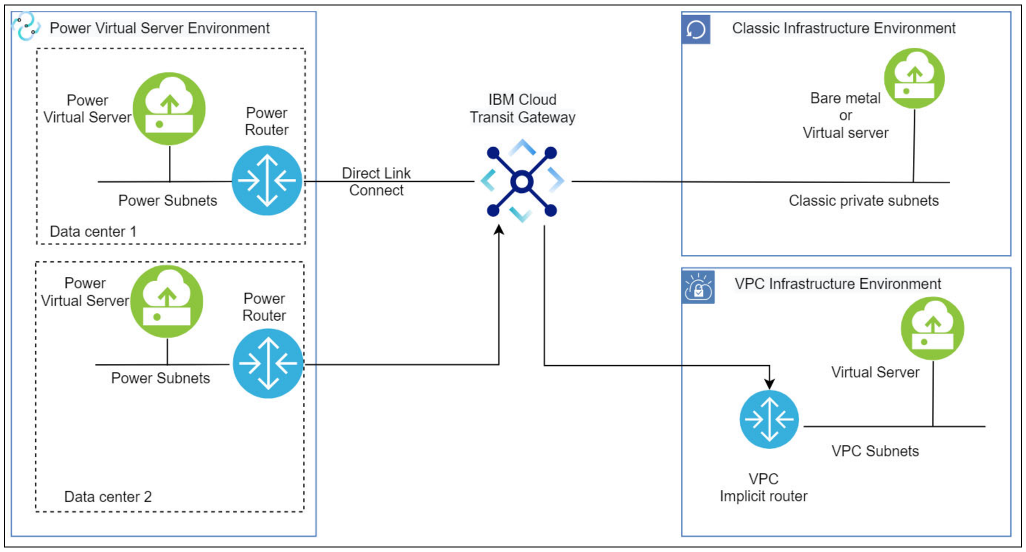


*Figure 4-11   Transit Gateway deployment scenario with DL2.0 or Megaport*

For more information, see Getting started with IBM Cloud Direct Link.

- **Power Edge Router:**

  A Power Edge Router (PER) is a high-performance router that provides advanced routing capabilities for IBM Power Virtual Server. One of the benefits of the PER solution is that it makes it easier for an IBM Power Systems Virtual Server instance to access other IBM Cloud services, such as IBM Cloud domain-name systems (DNS), Network Time Protocol (NTP, and Cloud Object Storage (COS). This allows connectivity to these services without having to use proxies or virtual routers, as the PER solution includes a Network Address Translation (NAT) device that simplifies the access process.

  Figure 4-12 illustrates the use of PER in IBM Cloud.

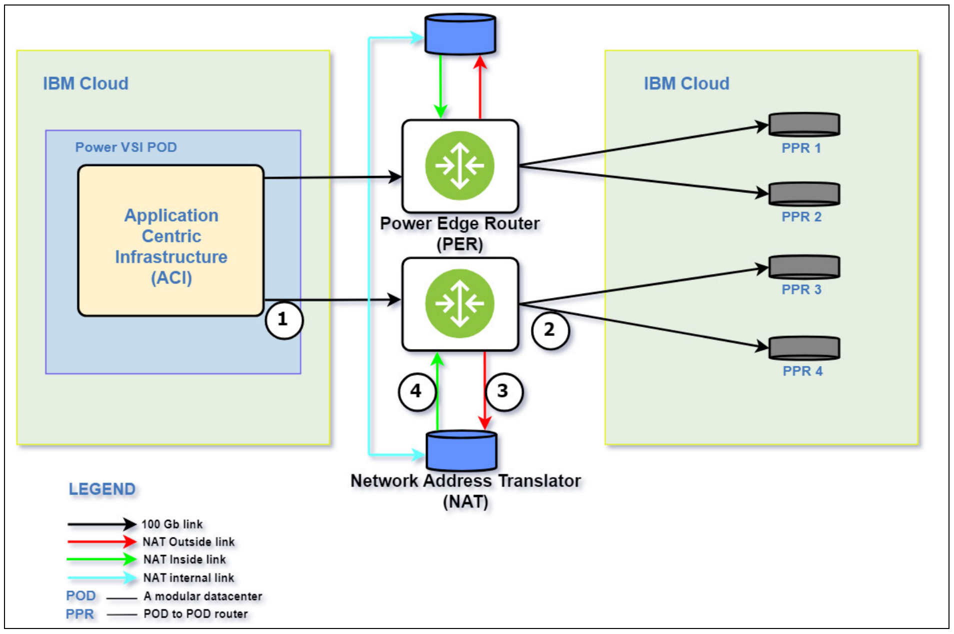


*Figure 4-12   Power Edge Router Integration to IBM Cloud*

PER improves network communication across different parts of the IBM network. The PER solution creates a direct connection to the IBM Cloud MPLS (Multi Protocol Label Switching) backbone, making it easy for different parts of the IBM network to communicate with each other.

**Note:** At the time of writing, the PER solution is currently only available in DAL10. PER will be deployed in other data centers over time.

For more information, see Power Virtual Server networking environment.

**Important:** Not all IBM Cloud locations provide the same external connectivity and this may depend on what is available depending on the available capabilities of local operators.

## 4.4 IBM Power Virtual Server storage options

This section explains various storage options for achieving high availability for Pacemaker clusters. Storage itself has a technology to achieve High availability through replication or IBM FlashCopy® features. HA with a Pacemaker cluster can be achieved in the cloud and on-premises.

Some storage options are GRS (Global Replication Services), GFS2 (Global file system 2), Shared Storage technology, and IBM Storage Scale.

### 4.4.1 GFS2

GFS2 is a shared disk file system for Linux clusters. GFS2 allows all the nodes in a cluster to have direct access to the same shared storage through the use of a locking protocol. All nodes in a GFS2 Linux cluster work as peer nodes.

Within RHEL, GFS2 is provided by the RHEL Resilient Storage Add-On, which requires the RHEL HA Add-On to provide the Pacemaker clustering option.

Using GFS2 in the cluster requires the correct hardware to allow access to shared storage and a locking mechanism to control access to the storage. The shared storage is usually provided by a SAN using Fibre Channel, NVMe over Fabrics, or iSCSI. The locking manager is built into your Linux distribution and requires the use of a cluster manager, such as Pacemaker.

For more information, see the GFS2 overview.

### 4.4.2 IBM Storage Scale

IBM Storage Scale is a high performance, HA, clustered file system and associated management software, available on different platforms. IBM Storage Scale can scale in several dimensions, including performance, capacity, and number of nodes that can mount or access the file system. IBM Storage Scale addresses the needs of applications whose performance-to-capacity ratio demands cannot be achieved by traditional scale-up storage systems. IBM Storage Scale therefore deploys for many I/O demanding enterprise applications that require high performance or scale. IBM Storage Scale provides various configuration options, access methods, and many other features such as snapshots, compression, and encryption.

For more information, see Power Virtual Server networking environment.

### 4.4.3 GRS

Data replication is key to business resiliency because simply put, data drives decision-making. Data informs and feeds into mission-critical processes, analytics, systems and, ultimately, business insights. Organizations must guarantee that data is constantly available and accessible to users in near real-time. As enterprises develop across geographies and platforms, replication enables them to scale in tandem with their expanding data requirements while maintaining performance. GRS reflects a commitment to enabling business continuity planning, operational excellence and optimization in cost with IBM Power Systems.

GRS is based on well-known, industry-standard IBM Storwize Global Mirror Change Volume Asynchronous replication technology. GRS on IBM Power Virtual Server exposes the API and CLI to create and manage replication-enabled volumes.

### 4.4.4 Configuration of HA Storage with Pacemaker

HA Storage with Pacemaker is a solution for ensuring that critical data and applications remain available and accessible even in the case of hardware failure or other disruptions. A Pacemaker cluster manager allows you to configure and manage clusters of computers that work together to provide HA for critical data and applications respectively.

The HA storage with Pacemaker solution typically involves clusters of multiple nodes, in which nodes have access to shared storage. This shared disk from the storage is used for storing critical application data, which can be accessed by any nodes of the cluster. In case of failure, the Pacemaker cluster automatically fails over to another node in the cluster, ensuring that the data and application remain available.

This section explains how to configure HA storage with Pacemaker on RHEL.

As a prerequisite, all the nodes of a cluster should have the appropriate shared disk from the same back-end storage.

1. Install Pacemaker on all nodes in the cluster using the command shown in Example 4-1.

*Example 4-1 Install Pacemaker*

```
$ sudo dnf install pacemaker pcs fence-agents-all
```

   This command installs the Pacemaker software, the `pcs` command line, and the collection of fencing agents.

2. Configure the firewall to allow access to the Pacemaker cluster on all nodes using the commands shown in Example 4-2.

*Example 4-2 Configure firewall*

```
$ sudo firewall -cmd - -permanent - - add- service=high-availability
$ sudo firewall -cmd - -reload
```

3. Create a Pacemaker cluster by running the command shown in Example 4-3 on one of the nodes.

*Example 4-3 Create Pacemaker cluster*

```
$ sudo pcs cluster setup - -name <cluster_name> node1 node2 node3
```

This command will configure the Pacemaker configuration files and generate the cluster configuration file that can be used to create the cluster on all nodes.

4. Start the Pacemaker cluster on all the nodes using the command shown in Example 4-4.

*Example 4-4 Start the Pacemaker cluster*

```
$ sudo pcs cluster start - -all
```

This command starts the cluster services on all the nodes.

5. Add the shared storage devices to the cluster by running the command in Example 4-5.

*Example 4-5 Add the shared storage devices to the cluster*

```
$ sudo pcs cluster cib  stonith_cfg
```

The above command will add the XML snippet to the Pacemaker configuration file (as shown in Example 4-6) to define the shared storage devices as the storage device to be used for the cluster fencing mechanism.

*Example 4-6 Modify Pacemaker config file to add storage devices*

```
<configuration>
     <fencing>
     <method name="sbd">
        <device class="stonith" name="sbd">
           <instance_attributes stonith-timeout="60"/>
        </device>
     </method>
     </fencing>
</configuration>
```

Save the file and close it.

6. Enable the sbd (Stonith Block Device) service on all nodes using the command:

*Example 4-7 Enable the sbd service*

```
$ sudo systemctl enabld sbd
```

7. Configure the Pacemaker cluster to use the shared storage device for HA storage by executing the command shown in Example 4-8.

*Example 4-8 Configure the Pacemaker cluster to use shared storage device for HA*

```
$ sudo pcs resource create shared_storage ocf:heartbeat:Filesystem device="/dev/sdb1" \
directory="/mnt/shared_storage" fstype="xfs" options="nofail" op monitor interval="30s" \
on-fail="standby" clone
```

The command will create a Pacemaker resource agent that monitors the shared disk and automatically fails over to another node in the Cluster if a problem is detected. Replace `/dev/sdb1` with the path to the required shared disk path and also change the mount according to the existing mount in your setup.

8. Verify that the shared disk is all available to the nodes of the cluster by running the command:

*Example 4-9 Verify that the shared disk is available to the nodes of the cluster*

```
$ sudo pcs status resources
```

   This command will display the status of all resources in the Pacemaker cluster, including the shared disks.

9. Test the HA storage configuration by simulating a node failure using the command shown in Example 4-10.

*Example 4-10 Test the HA storage configuration*

```
$ sudo pcs cluster simulate node1
```

   This command will fail `node1` and move the access to the other node.

For more information, see High Availability Storage with Pacemaker.

5

# HA for Database Servers

Pacemaker plays a pivotal role in ensuring continuous availability and reliability for critical services, including IBM Db2 databases. Leveraging Pacemaker for Db2 involves implementing a well-thought out strategy that maximizes uptime, minimizes downtime, and guarantees data integrity.

Pacemaker orchestrates the deployment and management of Db2 in a clustered environment, allowing seamless failover and resource allocation across multiple nodes. Its suite of failover policies, constraints, and monitoring capabilities empowers administrators to design and maintain a highly available database system that can withstand node failures, network interruptions, or planned maintenance events.

The integration of Pacemaker with Db2 introduces mechanisms for handling failovers, managing resource dependencies, and ensuring the efficient allocation of computing resources.

This chapter explores the intricacies of deploying and configuring Pacemaker to orchestrate HA for Db2 databases. It delves into failover policies, resource constraints, and best practices for setting up a fault-tolerant environment tailored specifically to Db2.

In this chapter we discuss the following:

## 5.1 Db2 database introduction

Db2 is a family of database management system (RDBMS) products from IBM that serves several different operating system (OS) platforms. Used by organizations of all sizes, Db2 provides a data platform for both transactional and analytical operations, as well as continuous availability of data to keep transactional workflows and analytics operating efficiently. In addition to being a relational DBMS, Db2 also offers integrated support for several NoSQL capabilities, including XML, graph store and JavaScript Object Notation (JSON).

### 5.1.1 Db2 database concepts

A Db2 database is a group of data treated collectively as a unit. A database is a large, structured set of persistent data, and its purpose is to store, retrieve and manipulate related information. The data is stored in tables. The Db2 DBMS operates as the server to manage data in databases across a multiuser environment, enabling many concurrent users to access the same data simultaneously. The Db2 DBMS also prevents unauthorized access, provides utilities for backing up and recovering data, and offers performance tools and data management capabilities.

Db2 databases have logical structures and physical structures that the DBMS manages separately. The physical storage of data can be managed without affecting the access to logical storage structures. Db2 databases are created using Data Definition Language (DDL) commands and are composed of tablespaces, tables with rows and columns, views, indexes, stored procedures, and other supporting structures. A database administrator (DBA) or developer can use a Db2 database and its underlying structures to create, read, update, and delete data to support an organization's business requirements. SQL Language is used by applications to interact with the database to perform create or change data.

### 5.1.2 Db2 HA and DR

HA/DR provides an HA solution to protect against both partial and complete site failures. HA/DR protects against data loss by replicating data changes from a source database, called the *primary database*, to the target databases, called the *standby databases*. HA/DR supports up to three remote standby servers.

A partial site failure can be caused by a hardware, network, or software failure. Without HA/DR, a partial site failure requires restarting the database management system (DBMS) server that contains the database. The length of time that it takes to restart the database and the server where it is located is unpredictable. It can take several minutes before the database is brought back to a consistent state and made available. With HA/DR, a standby database can take over in seconds. Furthermore, you can redirect the clients that used the original primary database to the new primary database by using automatic client reroute or retry logic in the application.

A complete site failure can occur when a disaster, such as a fire, causes the entire site to be destroyed. However, because HA/DR uses TCP/IP for communication between the primary and standby databases, they can be situated in different locations. For example, the primary database might be at your head office in one city, and a standby database might be at your sales office in another city. If a disaster occurs at the primary site, data availability is maintained by having the remote standby database take over as the primary database with full Db2 functionality. After a takeover operation occurs, you can bring the original primary database back up and return it to its primary database status; this procedure is known as *failback*. You can initiate a failback if you can make the old primary database consistent with the new primary database.

After you reintegrate the old primary database into the HA/DR setup as a standby database, you can switch the roles of the databases. This operation enables the original primary database to be the primary database again.

Figure 5-1 shows a Db2 database utilizing HA/DR. The example shows HA/DR deployed with one primary and two standby instances. One database instance in Data Center 1 is running in a primary role, while a standby database instance runs in Data Center 1 and a second standby database instance is available for DR in Data Center 2.

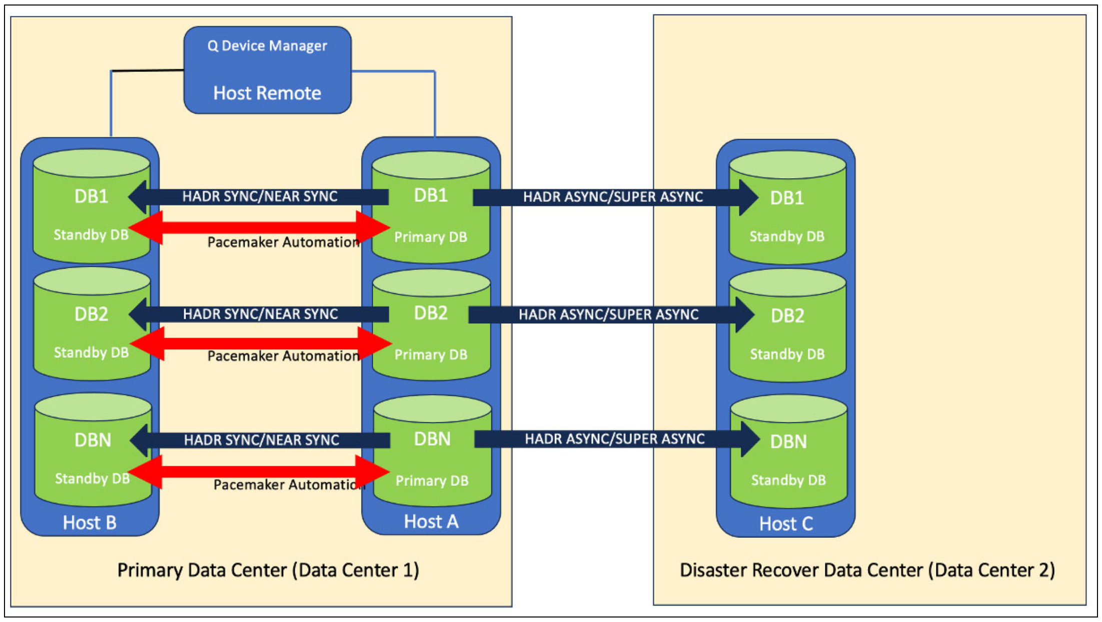


*Figure 5-1 DB2 HA/DR Architecture with Pacemaker across Data Centers*

## 5.1.3 Deploying Db2 databases with HA/DR

When Db2 database servers are deployed for HA, the database runs on multiple Db2 server hosts. Data is only allowed to be changed on one of the DB2 instances; the data is replicated to the other instances that are running in standby mode. The HA/DR feature of Db2 is responsible for managing the data replication. The normal implementation of Db2 HA/DR uses one primary Db2 instance and one standby DB2 instance in a production site. Optionally, another standby database runs in the DR site, if DR protection is desired.

For Db2 on Linux, Pacemaker, the Linux clustering component, is deployed to automate the database service availability. Additional requirements for Pacemaker are listed in section 5.4, “Prerequisites for an integrated solution using Pacemaker” on page 70. A cluster domain is created in the primary data center since it cannot span across the distance to the DR data center. Pacemaker will maintain and manage a Virtual IP address (VIP), which is used to access the current primary database instance. The application will use this VIP to connect to the database. The Pacemaker cluster manages the Db2 server’s HA/DR Role. The Db2 HA/DR feature manages data replication between the instances.

Figure 5-2 shows the components involved in running Db2 HA/DR database. Db2 with the HA/DR feature manages the data replication. Pacemaker ensures that if the primary Db2 instance fails, the secondary instance is promoted to primary and continues handling data requests.

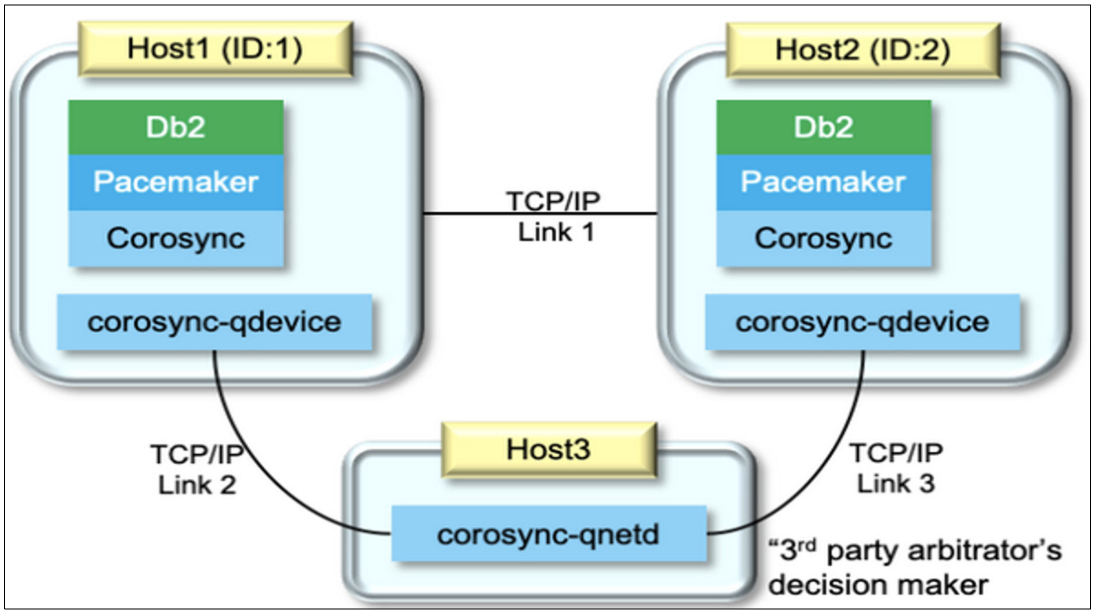


*Figure 5-2   DB2 Pacemaker typical high-level deployment architecture*

The DB2 database server is typically deployed in the following pattern:

- The primary data center with primary database and principal standby database is hosted on two database server hosts.
- Time synchronization needs to be maintained between the servers in different sites.
- The auxiliary standby database can be deployed in a standby data center.
- Service failover in the primary data center is automatically managed by Pacemaker. The server will fail over from the primary and standby database and is managed by Pacemaker.
- In the event of planned or unplanned downtime, the database service has to be manually started in a DR site, in case of failure.
- Allocate two network interface card (NIC) adapters on each production Db2 server to separate workload for client interactions and workload from database data replication.
  - NIC 1 for corporate application client connectivity.
  - NIC 2 for DB2 HA/DR replication network and connectivity.

## 5.2 Db2 high-level deployment architecture with Pacemaker

A cluster is a group of connected machines that work together as a single system. When one machine in a cluster fails, cluster management software transfers the workload of the failed machine onto other machines.

Pacemaker is an open-source HA cluster resource manager software that runs on a set of nodes. Together with Corosync (an open-source group communication system that provides ordered communication delivery, cluster membership, quorum enforcement, and other features among the nodes) it helps to detect component failures and orchestrate the necessary failover procedures to minimize interruptions to applications. Pacemaker provides a framework to manage the availability of resources (services on a host that need to be kept highly available).

DB2 Pacemaker utilizes Corosync to set up and manage the failover between the machines managed by the quorum arbitrator explained in section 5.3.2, “Quorum device support on Pacemaker” on page 69.

## 5.3 Using Pacemaker to create highly available Db2 databases

Pacemaker is seamlessly integrated with Db2 on Linux platforms, including Linux for IBM Power. The unified solution extends its support across multiple databases and multiple instances, all orchestrated through the Db2 cluster manager utility (db2cm).

Starting with Db2 11.5.5 and later, Pacemaker enables the automation of HA/DR configurations as the cluster manager. This advanced capability encompasses the automation of multiple HA/DR databases per instance and facilitates the automation of multiple HA/DR instances, all seamlessly coordinated within a shared Pacemaker/Corosync cluster environment.

Moreover, in Db2 11.5.8 and later, Pacemaker supports Mutual Failover (MF) automation. This integration empowers Pacemaker as the cluster manager to facilitate Mutual Failover, enhancing the DR capabilities of the system.

**Note:**

- In Db2 11.5.4, Pacemaker is provided solely as a technology preview, suitable for development, testing, and proof-of-concept purposes.
- Beginning with Db2 11.5.5, Pacemaker becomes part of the offering and is made accessible for deployment in production environments.
- In Db2 11.5.6 and later versions, the Pacemaker cluster manager is bundled and installed along with Db2 to facilitate automated failover to HA/DR standby databases.
- From Db2 11.5.8 and later, the utilization of Pacemaker as the integrated cluster manager allows for Mutual Failover HA.

### 5.3.1 Pacemaker core: foundational components and key elements

In the comprehensive HA setup integrating Pacemaker, the cluster software stack comprises multiple essential components, all vital for the effective operation of Pacemaker.

#### *Resources*

Resources represent specific entities within Db2 that require monitoring, initiation, or termination of state. This encompasses various elements (resources), such as the Db2 member process, HA/DR-capable databases for HA/DR, mount points necessary for Mutual Failover functionality, Ethernet network adapters, and virtual IP addresses.

### Constraints

Constraints serve as rules configured during cluster establishment to enhance the behavior of processes:

- Location Constraint: Determines where specific resources are permitted to operate within the cluster.
- Ordering Constraint: Specifies the sequence in which certain resource actions need to take place.
- Co-location Constraint: Defines the interdependency of one resource's location with another resource's placement within the cluster environment.

### Resource set

A collection of resources that fall under the influence of a specific constraint.

### Resource model

Within Pacemaker, the resource model for Db2 encompasses the predefined relationships and constraints of all resources. This model is established during cluster setup using the db2cm utility with the *-create* option. Any modifications or deviations from this model without approval from Db2 render the model unsupported.

### Resource agents

In Pacemaker, resource agents are Db2 user exits, consisting of a set of shell scripts developed and maintained by Db2. These agents execute actions on the resources defined within the resource model. Table 5-1 identifies available resource agents.

*Table 5-1 Resource agents*

| Resource agent | Definition | Level |
|---|---|---|
| db2ethmon | The resource agent to monitor the defined Ethernet network adapter. | Host level. |
| db2inst<br>(HA/DR only) | The resource agent to monitor, start, and stop the Db2 member process. | Db2 instance level. |
| db2HA/DR<br>(HA/DR only) | The resource agent to monitor, start, and stop individual HA/DR-enabled databases. | Db2 database level. |
| db2partition<br>(Mutual Failover) | The resource agent to monitor, start, and stop the Db2 partition process. | Db2 instance level. |
| db2fs<br>(Mutual Failover) | The resource agent to monitor, start, and stop a file system. | N/A. |

### Cluster topology and communication layer

Effective HA cluster management requires consistent cluster topology among nodes. Pacemaker employs the Corosync Cluster Engine, an open-source group communication system, to ensure uniform cluster topology across nodes. Corosync guarantees a reliable messaging infrastructure, ensuring that events occur in the same sequence on each node, and enforces quorum constraints.

### Cluster Domain Leader

Within the cluster, one node is elected as the Domain Leader (also known as the Designated Controller or DC in Pacemaker terminology). The Pacemaker controller daemon on the Domain Leader assumes responsibility for making critical cluster decisions. If the current Domain Leader node fails, a new Domain Leader is elected to assume this pivotal role.

#### *Networks*

To set up components within your Pacemaker cluster domain associated with networking, use the Db2 cluster manager (db2cm) utility. This utility allows the addition of a physical network to your cluster domain. This physical network typically consists of network interface cards (NICs), IP addresses, and subnet masks.

#### *QDevice*

The QDevice (Quorum Device) refers to a specialized quorum configuration that utilizes an external device or mechanism to determine the cluster's availability and make decisions regarding its operation. QDevice quorum provides an additional layer of reliability by introducing an external resource (such as a storage device or network device) to participate in the cluster's quorum. Key aspects of QDevice quorum include:

- QDevice quorum involves integrating an external device or mechanism into the cluster's quorum calculations. This device acts as an additional voter in determining the cluster's state and making decisions regarding failover or resource management.
- By adding an external quorum device, the cluster's quorum decisions are less reliant on the number of cluster nodes. Instead, the device's availability or state contributes to the overall quorum calculation, enhancing reliability and reducing the risk of split-brain scenarios or incorrect failover decisions.
- Quorum devices can vary and might include shared disks, dedicated quorum servers, or network-based devices specifically designed to participate in quorum calculations. These devices are configured to communicate with the cluster nodes and provide input to determine the cluster's operational state.
- The QDevice's availability contributes to the quorum vote count, influencing the cluster's ability to continue operations or trigger failover actions. If the external device is accessible and operational, it adds its quorum vote to the overall decision-making process.
- In scenarios where the majority of cluster nodes become unavailable or network partitions occur, the presence and operational status of the QDevice can be decisive in making proper failover decisions to ensure cluster integrity and data consistency.

**Note:** QDevice quorum is the recommended quorum mechanism for a production system.

## 5.3.2 Quorum device support on Pacemaker

A quorum device helps a cluster manager make cluster management decisions when the cluster manager's normal decision process does not produce a clear choice. To select an action to take, a cluster manager counts the number of cluster domain nodes that support each of the potential actions. The cluster manager then selects the action that is supported by most of the cluster domain nodes. If the same number of cluster domain nodes supports more than one choice, then the cluster manager refers to a quorum device to make the choice. The Pacemaker cluster stack supports the following quorum devices:

- Two-node quorum
- Majority quorum
- QDevice quorum

### Two-node Quorum and Majority Quorum

The two-node quorum is the default mechanism. Because no tie-breaker mechanism exists, the two-node quorum is prone to the split-brain scenario. It is not intended for production environments.

Majority quorum is intended to use with cloud providers such as AWS, GCP, or AZURE.

### QDevice quorum

This is the recommended quorum management method for Db2 HA/DR. QDevice requires an external resource that must be accessible by all hosts in the Pacemaker cluster. QDevice is much more reliable because the quorum decision logic is more robust than a simple TCP/IP ping to the external IP address.

QDevice requires that the resource is placed on a separate host, like a majority quorum requirement. However, setup is simplified, as the host with the QDevice does not need to be part of the Pacemaker quorum. The QDevice is an excellent choice as a quorum solution as it combines reliability and simplicity.

## 5.4 Prerequisites for an integrated solution using Pacemaker

The following are the prerequisites for an integrated solution using Pacemaker for Db2 on IBM Power Linux. For more information, see Prerequisites for an integrated solution using Pacemaker.

### *Supported IBM Power Linux distributions*

Table 5-2 identifies the supported Power Linux distributions for running a Pacemaker-managed integrated HA solution.

*Table 5-2 Supported Power Linux for Db2*

| Db2 version | Supported RHEL version | Supported SUSE Version |
|---|---|---|
| Db2 11.5.8 | Version 8.4 and higher | 15 SP3 and above |
| Db2 11.5.7 | Version 8.2 and higher | 15 SP3 and above |
| Db2 before 11.5.7 | Not supported | Not supported |

### *User Configuration*

Ensure that the necessary users (such as instance owner and fenced user), associated groups, and other users essential for the setup are created.

### *SSH Configuration*

Set up passwordless SSH between HA/DR nodes for both root and instance users to enable seamless communication.

### *Storage Requirements*

Allocate adequate local storage on each node for cluster-related software:

– Approximately 50 MB for cluster storage RPMs and extracted files.
– Around 200 MB for a full installation of cluster-related software.
– Minimum 1 GB space in `/var` to store cluster software log files.
– At least 150 MB in /usr for RHEL and 300 MB for SUSE distributions.

### *RPM Requirements*

Please enable a Yum repository for installation, which looks like:

- `/usr/share/pacemaker`
- `/usr/share/doc/packages`
- `/usr/share/licenses`
- `/usr/share/man/man7`
- `/usr/share/man/man8`
- `/usr/lib/pacemaker`
- `/usr/lib/ocf/resource.d/pacemaker`
- `/usr/lib/systemd/system`
- `/usr/lib/debug/dwz`
- `/usr/lib64`
- `/usr/lib64/pkgconfig`
- `/usr/sbin`

### *Firewall configuration*

If a firewall is in place, open specific ports (`3121`, `5403`, `5404` - `5405`) on each host or in the network to facilitate communication.

### *Required packages*

Ensure the presence of KornShell (*ksh*) and *python3-dnf-plugin-versionlock* packages for Pacemaker. The latter is used by the Db2 installer to lock all relevant Pacemaker and Corosync RPMs.

### *Virtual IP setup*

In Db2 HA/DR, set up a Virtual IP for each HA/DR-enabled database to enable automatic client rerouting during failover.

### *QDevice quorum*

To establish QDevice quorum, install corosync-qnetd software on a third host acting as the arbitrator. This host does not need the Db2 server and serves as a mediator.

### *The db2prereqcheck command*

Executing this command allows you to check if your system meets the prerequisites prior to commencing the installation. Remember, you need to download the installation media and run db2prereqcheck for Db2 to proceed with the installation.

## 5.5 Installing Db2 on RHEL and configuring Pacemaker

The following are the required steps to install and configure Db2 with Pacemaker on RHEL:

- Download the latest version of Db2 from IBM Passport Advantage® Online for Customers web site or IBM PartnerPlus Software Access Catalog.
- Ensure that you have the supported version of RHEL, as described on Table 5-2 on page 70.

- Extract the archive and unpack each of the parts into the working directory that you created on your two LPARs where you will install the Primary and Standby Db2 databases. The archive will contain the following directory tree:
  - `Db2/`
  - `Db2agents/`
  - `RPMS/`
  - `RPMS/ppcle`
  - `RPMS/noarch`
  - `SRPMS/`
- To check whether the system meets the prerequisites for Db2 11.5.8, issue the command shown in Example 5-1.

*Example 5-1 db2prereqcheck command*

```
$ db2prereqcheck -v 11.5.8.0 -s
```

- Install Pacemaker using the db2_install or db2setup[1] command: you can run the commands without any parameters specific to Pacemaker. The default behavior is to install Pacemaker. Set up Db2 according to the Db2 11.5 documentation.

## 5.6 Using the Db2 cluster manager utility for HA

The Db2 cluster manager (db2cm) tool facilitates the setup and management of robust Db2 databases within a Pacemaker cluster to ensure HA. Here are two key configurations available:

- Use the HA/DR HA feature within a Db2 instance operating on a Pacemaker-controlled Linux cluster.
- Use the Mutual Failover HA capability within a Db2 instance running on a Pacemaker-controlled Linux cluster.

### 5.6.1 HA/DR for Pacemaker-managed Db2 instances

HA/DR offers a solution for maintaining operational continuity during both partial and complete site outages. It safeguards against data loss by replicating data modifications from a source database (the primary) to designated databases (the standbys). HA/DR facilitates up to three remote standby servers to enhance resilience.

Prior to setting up HA/DR, confirm the presence of the subsequent system prerequisites:

- All hosts within the cluster must have the Pacemaker cluster software stack installed.
- Configure and ensure the online status of Db2 instances and the HA/DR-enabled database.
- In this scenario, `192.168.1.100` and `192.168.1.200` represent the host names of the cluster nodes, `yslhadomain` stands as the domain name, and `eth0` serves as the network interface (device) name allocated to each host.
- Create the Pacemaker cluster and the public network resources by running the command shown in Example 5-2 on page 73.

[1] The db2setup utility, which provides silent install with a response file and a graphical user interface, supports installing Pacemaker in Db2 11.5.8 and later.

*Example 5-2 Create Pacemaker cluster command*

```
$ home/db2inst1/sqllib/bin/db2cm -create -cluster -domain yslhadomain -host
ip-192.168.1.100 -publicEthernet eth0 -host ip-192.168.1.200 -publicEthernet eth0
Created db2_ip-192.168.1.100_eth0 resource.
Created db2_ip-192.168.1.200_eth0 resource.
Cluster created successfully.
```

- Create the instance cluster domain by running the commands shown in Example 5-3.

*Example 5-3 Create instance cluster domain command*

```
$/home/db2inst1/sqllib/bin/db2cm -create -instance db2inst1 –host ip-192.168.1.100
Created db2_ip-192.168.1.100_db2inst1_0 resource.
Instance resource for db2inst1 on ip-192.168.1.100 created successfully.

$ /home/db2inst1/sqllib/bin/db2cm -create -instance db2inst1 –host ip-192.168.1.200
Created db2_ip-192.168.1.200_db2inst1_0 resource.
Instance resource for db2inst1 on ip-192.168.1.200 created successfully.
```

- Verify the cluster by running the crm status command as shown in Example 5-4.

*Example 5-4 Verify cluster status command*

```
$crm status
Stack: corosync
Current DC: ip-192.168.1.200 (version 2.0.2-1.el8-744a30d655) - partition with quorum
Last updated: Tue Dec 05 21:49:57 2023
Last change: Tue Dec 05 21:39:45 2023 by root via cibadmin on ip-192.168.1.100
2 nodes configured
4 resources configured
Online: [ ip-192.168.1.200 ip-192.168.1.100 ]
Full list of resources:
db2_ip-192.168.1.100_eth0 (ocf::heartbeat:db2ethmon): Started ip-192.168.1.100
db2_ip-192.168.1.200_eth0 (ocf::heartbeat:db2ethmon): Started ip-192.168.1.200
db2_ip-192.168.1.100_db2inst1_0 (ocf::heartbeat:db2inst): Started ip-192.168.1.100
db2_ip-192.168.1.200_db2inst1_0 (ocf::heartbeat:db2inst): Started ip-192.168.1.200
```

- Create a new database and configure HA/DR on the new database.
- Create the HA/DR database resources as shown in Example 5-5.

*Example 5-5 Create HA/DR database resources*

```
$/home/db2inst1/sqllib/bin/db2cm -create -db YSLDB -instance db2inst1
Database resource for YSLDB created successfully
```

- Create the VIP resources for the newly created database, as shown in Example 5-6.

*Example 5-6 Create VIP resources*

```
$/home/db2inst1/sqllib/bin/db2cm -create -primaryVIP 192.168.1.201 -db YSLDB –instance
db2inst1
```

- Verify the cluster by running the crm status command after creating Db2 databases, HA/DR resources, and VIP resources, as shown in Example 5-7.

*Example 5-7 Verify cluster status*

```
$ crm status
Stack: corosync
Current DC: ip-192.168.1.200 (version 2.0.2-1.el8-744a30d655) - partition with quorum
Last updated: Tue Dec 05 22:20:52 2023
Last change: Tue Dec 05 22:14:29 2023 by root via cibadmin on ip-192.168.1.200
2 nodes configured
7 resources configured
Online: [ ip-192.168.1.200 ip-192.168.1.100 ]
Full list of resources:
db2_ip-192.168.1.100_eth0 (ocf::heartbeat:db2ethmon): Started ip-192.168.1.100
db2_ip-192.168.1.200_eth0 (ocf::heartbeat:db2ethmon): Started ip-192.168.1.200
db2_ip-192.168.1.100_db2inst1_0 (ocf::heartbeat:db2inst): Started ip-192.168.1.100
db2_ip-192.168.1.200_db2inst1_0 (ocf::heartbeat:db2inst): Started ip-192.168.1.200 Clone
Set: db2_db2inst1_db2inst2_YSLDB-clone [db2_db2inst1_db2inst2_YSLDB] (promotable)
Masters: [ ip-192.168.1.200 ]
Slaves: [ ip-192.168.1.100 ]
db2_db2inst1_db2inst1_YSLDB-primary-VIP (ocf::heartbeat:IPaddr2): Started ip-192.168.1.200
```

- Install and configure a QDevice quorum.
  - On the primary and standby hosts, install the corosync-qdevice packages, as shown in Example 5-8.

*Example 5-8 Install the corosync-qdevice packages*

```
$dnf install
<Db2_image>/db2/<platform>/pcmk/Linux/<OS_distribution>/<architecture>/corosync-qdevice*
```

  - On the QDevice host, install the Corosync QNet software, as shown in Example 5-9.

*Example 5-9 Install Corosync QNet software*

```
$dnf install
<Db2_image>/db2/<platform>/pcmk/Linux/<OS_distribution>/<architecture>/corosync-qnetd*
```

  - As the root user, run the db2cm command to set up the QDevice from one of the cluster nodes, as shown in Example 5-10.

*Example 5-10 Setup QDevice*

```
$ <Db2_image>/bin/db2cm -create -qdevice qserver.powerm.ma
```

  - On the primary and standby hosts, verify that the quorum was set up correctly, as shown in Example 5-11.

*Example 5-11 Verify Quorum Setup*

```
$corosync-qdevice-tool -s
Qdevice information -------------------
Model:           Net
Node ID:        1
Configured node list:
0    Node ID = 1
1    Node ID = 2
Membership node list:    1, 2
```

```
Qdevice-net information ----------------------
Cluster name:     yslhadomain
QNetd host:       qserver.powerm.ma:5403
Algorithm:        LMS
Tie-breaker:      Node with lowest node ID
State:            Connected
```

– On the QDevice host, verify that the quorum device is running correctly, as shown in Example 5-12.

*Example 5-12 Verify Quorum*

```
$ corosync-qnetd-tool -l
```

- Initiate the takeover of a primary HA/DR host from the standby host on a Pacemaker-managed Db2 instance.
  – Verify cluster status, as shown in Example 5-13.

*Example 5-13 Verify Cluster*

```
$ crm status
Stack: corosync
Current DC: dbsecondary (version 2.0.2-1.el8-744a30d655) - partition with quorum
Last updated: Tue Dec 05 22:22:32 2023
Last change: Tue Dec 05 22:15:43 2023 by root via cibadmin on dbprimary
2 nodes configured
7 resources configured
Online: [ dbprimary dbsecondary ]
Full list of resources:
db2_dbprimary_eth1 (ocf::heartbeat:db2ethmon): Started dbprimary db2_dbsecondary_eth1
(ocf::heartbeat:db2ethmon): Started dbsecondary db2_dbprimary_db2inst1_0
(ocf::heartbeat:db2inst): Started dbprimary db2_dbsecondary_db2inst1_0
(ocf::heartbeat:db2inst): Started dbsecondary
Clone Set: db2_db2inst1_db2inst1_YSLDB-clone [db2_db2inst1_db2inst1_YSLDB] (promotable)
Masters: [ dbprimary ]
Slaves: [ dbsecondary ]
db2_db2inst1_db2inst1_YSLDB-primary-VIP (ocf::heartbeat:IPaddr2): Started dbprimary
db2_db2inst1_db2inst1_YSLDB-standby-VIP (ocf::heartbeat:IPaddr2): Started
dbsecondary
```

– As the Db2 instance owner, run a user-initiated takeover from the standby host as shown in Example 5-14.

*Example 5-14 Initiate takeover*

```
$ db2 takeover HA/DR on db YSLDB
DB20000I The TAKEOVER HA/DR ON DATABASE command completed
successfully.
```

– Validate that the primary role fails over in the crm output. Also, verify that the reads on the standby VIP and primary VIP have switched hosts as expected. This can be seen in Example 5-15 on page 76.

*Example 5-15 Verify Cluster*

```
$ crm status
Stack: corosync
Current DC: dbsecondary (version 2.0.2-1.el8-744a30d655) - partition with quorum Last
updated: Tue Dec 05 22:26:42 2023
Last change: Tue Dec 05 22:17:35 2023 by root via cibadmin on dbprimary
2 nodes configured
7 resources configured
Online: [ dbprimary dbsecondary ]
Full list of resources:
db2_dbprimary_eth1      (ocf::heartbeat:db2ethmon):     Started dbprimary
db2_dbsecondary_eth1       (ocf::heartbeat:db2ethmon):     Started dbsecondary
db2_dbprimary_db2inst1_0       (ocf::heartbeat:db2inst):       Started dbprimary
db2_dbsecondary_db2inst1_0  (ocf::heartbeat:db2inst):       Started dbsecondary Clone Set:
db2_db2inst1_db2inst1_YSLDB-clone [db2_db2inst1_db2inst1_YSLDB] (promotable)
Masters: [ dbsecondary ]
Slaves: [ dbprimary ]
db2_db2inst1_db2inst1_YSLDB-primary-VIP (ocf::heartbeat:IPaddr2): Started dbsecondary
db2_db2inst1_db2inst1_YSDB-standby-VIP (ocf::heartbeat:IPaddr2): Started
dbprimary
```

For more information, see Maintaining a Pacemaker-managed cluster domain for an HADR Db2 instance.

## 5.6.2 Mutual Failover for Pacemaker-managed Db2 instances

Db2 versions 11.5.8 and beyond offer an alternative HA feature known as Mutual Failover when utilized within a Pacemaker-controlled Linux cluster. The setup of this cluster is established using the Db2 cluster manager (*db2cm*) utility.

Db2 Mutual Failover represents a HA configuration where two hosts possess the same Db2 instance installation and a shared mount, allowing only one host to be active at any given moment

In this scenario, `192.168.1.100` and `192.168.1.200` represent the host names of the cluster nodes, `yslhadomain` stands for the domain name, and `eth0` serves as the network interface (device) name allocated to each host.

- On host `192.168.1.100`:
  - As the root user, create a directory on `192.168.1.100` for the file system that is to be shared by the two Db2 Mutual Failover hosts (`192.168.1.100` and `192.168.1.200`), as shown in Example 5-16.

*Example 5-16 Create a directory for the shared file system.*

```
$mkdir /db2ha
```

  - Format a partition on the target device with the appropriate file system for the cluster, as shown in Example 5-17.

*Example 5-17 Format a partition.*

```
$mkfs.ext3 /dev/sda
```

– Get the Universal Unique Identifier (UUID) of the mount device that you create, as seen in Example 5-18.

*Example 5-18 Retrieve UUID.*

```
$blkid /dev/sda1
...
UUID="31a28d09-381c-4249-8b88-1d5db75c3b81" BLOCK_SIZE="4096" TYPE="ext3"
```

– Create a new entry in `/etc/fstab` on both hosts, as shown in Example 5-19.

*Example 5-19 Create a fstab entry.*

```
$/dev/sda1:UUID=31a28d09-381c-4249-8b88-1d5db75c3b81 /db2ha   ext3   acl,user_xattr,noauto
0 0
```

– Mount the new file system on this host, as shown in Example 5-20.

*Example 5-20 Mount a file system.*

```
$mount /db2ha
```

– Create administration groups, an instance user, and a fenced user on both hosts in your cluster, as shown in Example 5-21.

*Example 5-21 Create group and users.*

```
groupadd -g 990 db2iadm1
groupadd -g 989 db2fadm1
useradd -u 1002 -g db2iadm1 -m -d /home/db2inst1 -s /usr/bin/ksh db2inst1
useradd -u 1003 -g db2fadm1 -m -d /home/db2fenc1 -s /usr/bin/ksh db2fenc1
```

– Install Db2 on both hosts in your cluster, as shown in Example 5-22.

*Example 5-22 Install Db2.*

```
$./db2_install -b /opt/ibm/db2/V115M8A -p SERVER -y
```

– Create an empty directory for the db2inst1 user and copy the existing `.profile` and `.kshrc` files over to the folder, as shown in Example 5-23.

*Example 5-23 Copy files.*

```
mkdir /db2ha/db2home
chown -R db2inst1.db2iadm1 /db2ha
usermod -d /db2ha/db2home db2inst1
cp -f /home/db2inst1/.profile /db2ha/db2home
cp -f /home/db2inst1/.kshrc /db2ha/db2home
```

– Create the Db2 instance, as shown in Example 5-24.

*Example 5-24 Create a Db2 instance.*

```
$/opt/ibm/db2/V115M8A/instance/db2icrt -u db2fenc1 db2inst1
```

- Configure passwordless SSH by first adding the Db2 instance user's SSH key to the `authorized_keys` file on `192.168.1.100`, as shown in Example 5-25.

*Example 5-25 Configure passwordless SSH.*

```
$cat /db2ha/db2home>/.ssh/id_rsa.pub >> /db2ha/db2home /.ssh/authorized_keys
```

- On the second host 192.168.1.200.
  - Unmount the shared file system from the first host using the command shown in Example 5-26.

*Example 5-26 Unmount the file system.*

```
$umount /db2ha
```

  - Create a file system directory on the second host and then mount the shared file system to the directory. See Example 5-27.

*Example 5-27 Create a directory and mount the file system.*

```
$mkdir /db2ha
$ mount /db2ha
```

  - Confirm that the users are the same on both hosts, mount the shared file system and add the Db2 instance user's SSH key to `authorized_keys`, and configure the home directory on the second host and then remove all files from `sqllib`, as shown in Example 5-28.

*Example 5-28 Configure the home directory and remove all files from sqllib directory.*

```
chown -R db2inst1.db2iadm1 /db2ha
usermod -d /db2ha/db2home db2inst1
rm -rf /db2ha/db2home/sqllib
```

  - Create the Db2 instance on the second host using the command in Example 5-29.

*Example 5-29 Create a Db2 instance.*

```
$/opt/ibm/db2/V115M8A/instance/db2icrt -u db2fenc1 db2inst1
```

- Configure Mutual Failover with the db2cm command:
  - Run *db2cm* to create your Db2 cluster as seen in Example 5-30.

*Example 5-30 Create a Db2 cluster.*

```
$ home/db2inst1/sqllib/bin/db2cm -create -cluster -domain yslhadomain -host
ip-192.168.1.100 -publicEthernet eth0 -host ip-192.168.1.200 -publicEthernet eth0
Created db2_ip-192.168.1.100_eth0 resource.
Created db2_ip-192.168.1.200_eth0 resource.
Cluster created successfully.
```

  - Create a partition resource on the first host as shown in Example 5-31.

*Example 5-31 Configure the partition resource.*

```
$./db2cm -create -partition 0 -instance db2inst1
```

- Create a VIP resource on the first host, as shown in Example 5-32.

*Example 5-32 Create a VIP resource.*

```
$./db2cm -create -primaryVIP 192.168.1.201 -partition 0 -instance db2inst1
```

- Create a Quorum Device on the first host, as shown in Example 5-33.

*Example 5-33 Create a Quorum Device.*

```
$./db2cm -create -qdevice qserver
```

- Verify the configuration of your Mutual Failover, Pacemaker-managed Db2 cluster using the db2cm command, as shown in Example 5-34.

*Example 5-34 Verify configuration*

```
$./db2cm -list
```

- Initiate a failover and move a mutual failover partition from one host to the other.
  - As the root user, find the location of the partition that you want to move, as shown in Example 5-35.

*Example 5-35 Find partition location*

```
$./db2cm -list
```

  - As the root user, run db2cm -move to move your partition, from the install path, as shown in Example 5-36.

*Example 5-36 Move the partition.*

```
$/opt/ibm/db2/V11.5/bin/db2cm -move -instance db2inst1 -partition 0 -host dbsecondary
Partition resource db2_db2inst1_0 moved to dbsecondary successfully.
```

For more information, see Maintaining a Pacemaker-managed cluster domain for a Mutual Failover Db2 instance.

6

# Using Pacemaker for Highly Available SAP Landscapes

This chapter describes the use of Pacemaker on IBM Power Systems Virtual Server and on IBM Power Systems to provide HA (HA) SAP landscapes.

The two primary application areas of SAP used with IBM Power Systems include IBM Cloud and on-premises with Pacemaker. Specifically, HA deployments of SAP High-Performance Analytical Appliance (HANA) and HA SAP NetWeaver and S/4HANA applications are commonly implemented using IBM Power Systems. This setup allows for enhanced performance and availability for mission-critical SAP workloads.

The following sections will explore various approaches for deploying SAP HANA databases and SAP NetWeaver or S/4HANA applications on Power Systems:

## 6.1 Introduction to SAP

SAP offers solutions tailored to a wide range of businesses, from small enterprises to large corporations. Its primary focus is on centralizing business processes to provide real-time insights, enabling organizations to streamline and design their internal processes effectively. SAP serves over 25 industries with software solutions that enhance data management, decision-making, and planning, helping businesses achieve their goals and optimize their operations.

We present a summary of SAP products in this section. For more information, see What is SAP?.

### SAP ECC

SAP ECC, also known as SAP ERP or NetWeaver, is SAP's legacy suite of enterprise applications, designed to operate on third-party databases like Oracle or DB2. As the central component of the SAP Business Suite, ECC offers a comprehensive and integrated view of an organization's business processes, covering areas such as finance, warehousing, human resources, and logistics.

To appreciate ECC's broad impact, consider its range of modules, including:

- Financial Accounting and Controlling (commonly known as FICO)
- Sales and Distribution / Customer Service
- Materials Management
- Production Planning
- Quality Management
- Warehousing
- Logistics
- Plant Maintenance
- Human Resources Management

SAP Business Suite further enhanced its capabilities with advanced modules that integrate with ECC, such as Customer Relationship Management (CRM), Supplier Relationship Management, Supply Chain Management, and Product Life cycle Management.

**Important:** In 2020, SAP announced the end of maintenance support for its ECC product, beginning in 2027, an extension of the original 2025 date. Any new SAP implementations should be done on SAP S/4HANA. Customers running SAP ECC should be planning migration to SAP S/4HANA to maintain support.

### SAP Business Warehouse

SAP Business Warehouse (BW), offers a model-driven approach to enterprise data warehousing, simplifying and streamlining the process, especially for data from SAP R/3. Over the years, SAP BW has become an essential tool for thousands of organizations.

SAP BW is designed to transform and consolidate business information from nearly any source. When first introduced SAP BW operated on standard relational databases such as Oracle and IBM DB2. However, starting with version 7.4, it began transitioning to SAP HANA, SAP's in-memory database manager, enhancing its performance and capabilities.

### SAP HANA and SAP S/4HANA

SAP HANA is a revolutionary multi-modal database that stores data in memory rather than on disk. Introduced in 2010, this column-oriented, in-memory design was built to support both high-speed transactions and advanced analytics within a single system.

Because of its in-memory technology, SAP HANA can process vast amounts of data with near-zero latency This allows companies to quickly query data and leverage real-time insights,.

SAP HANA's architecture, which combines online analytical processing (OLAP) and online transactional processing (OLTP) in column-based tables stored in main memory, sets it apart from other database management systems (DBMS) available today. Initially, SAP HANA was designed to integrate with the legacy SAP ECC system, as an option to the previously supported databases.

In 2015, SAP introduced its next-generation ERP software, SAP S/4HANA. This suite is engineered to deliver real-time analytics, a streamlined user interface, and a simplified data model, aiming to enhance business efficiency and effectiveness. Featuring the SAP Fiori user experience, S/4HANA offers faster response times and performance improvements. Built on the advanced SAP HANA in-memory database, S/4HANA processes large volumes of data swiftly, enabling more informed decision-making through real-time insights. Unlike SAP ECC, which operated on third-party databases, S/4HANA is tightly integrated with SAP HANA.

S/4HANA integrates key business functions – including financial management, procurement, supply chain management, sales and distribution, production planning and control, and project management – into a unified system. It also serves as a crucial foundation for leveraging next-generation technologies such as robotic process automation, machine learning, big data management, the Internet of Things (IoT), and artificial intelligence (AI). S/4HANA is the base of SAP's intelligent enterprise model, driving productivity, cost reduction, and agility for quick response to changing business environments.

In 2016, SAP launched SAP BW/4HANA, an upgraded version of its enterprise data warehouse solution. Like SAP S/4HANA, SAP BW/4HANA is optimized for the SAP HANA database and operates exclusively on this platform.

## 6.1.1 HA concepts for SAP solutions

SAP ERP solutions consist of various interconnected components and services that create a cohesive network of functions. These components include:

- ABAP Central Services (ASCS)
- Java Central Services (SCS)
- Enqueue Replication Server (ERS)
- Primary Application Server (PAS)

In addition to these elements, the SAP landscape depends on a database, such as SAP HANA, which also requires HA protection. Your SAP environment may also incorporate other systems like NetWeaver and S/4HANA.

For an HA solution to be effective, it must continuously monitor the health of all SAP services, whether running databases or application server and including production and non-production instances. If a service underperforms or fails, the HA solution should promptly take action to recover the service and bring it back online as quickly as possible. It must ensure that services are restored on secondary nodes in the correct sequence to resume SAP functions efficiently and prevent data corruption.

Service restarts can occur on the same node or on an alternative node within the cluster. To facilitate restarting on an alternative node, the clustering software configures secondary nodes with standby services ready to take over if the primary node encounters an issue.

To meet strict recovery point objectives (RPOs), the failover cluster should be set up so that primary and secondary nodes either share the same storage (such as SAN) or have mirrored storage with efficient replication. Mirrored storage offers significant advantages: it facilitates failover clustering in cloud environments where shared storage may not be feasible and allows the use of local solid-state drives (SSD) for cost-effective, high-performance storage.

Failover clustering ensures minimal service downtime and reduces impact on end-users. Additionally, data replication protects the database by maintaining a copy on the secondary node within the same HA cluster, thus avoiding a single point of failure with the SAN. This data replication applies to various databases, including MaxDB, DB2, and SAP HANA. It's also crucial to geographically separate cluster nodes to protect against site-wide or regional disasters. Employ a clustering solution that supports failover between nodes in different cloud subnets or geographically distinct locations.

## 6.2 Using Pacemaker for S/4HANA or NetWeaver application resources

There are two primary deployment options for running SAP systems on IBM Power:

- On-premises: Utilize physical IBM Power systems hardware within your own data center.
- Cloud-based: Leverage IBM Power servers hosted on the IBM Cloud through the IBM Power Systems Virtual Server offering.

Running SAP NetWeaver or S/4HANA on IBM Power Systems, whether on-premises or through IBM Cloud PowerVS, provides a robust and consistent platform for SAP applications. This setup ensures top-tier performance, resilience for critical workloads, and a flexible infrastructure. Both on-premises hardware and IBM PowerVS are certified by SAP and support SUSE Linux and Red Hat Linux distributions.

The Pacemaker (HA Add-on Cluster) implementation is fully certified for SAP systems on IBM Power hardware. While the implementation is fundamentally the same for both on-premises and IBM PowerVS environments, there are minor differences in the commands used for each. Additionally, there are slight variations in the implementation process depending on whether SUSE Linux or Red Hat Linux is used.

**Note:** There are some minor differences when using Red Hat or SUSE Enterprise Linux and relevant OS vendor documentation must be referenced. The SAP notes will provide specific details relating to the implementation differences between the two operating systems.

Refer to the following SAP Notes for further details on how to plan, deploy and configure these options on IBM Power Systems:

– *SAP Note - 2378874 - Install SAP Solutions on Linux on IBM Power Systems (little endian)*
– *SAP Note - 2855850 - SAP Applications on IBM Power Virtual Servers*
– *SAP Note - 2923984 - SAP on IBM Power Virtual Servers: Support prerequisites*
– *SAP Note - 2772999 - Red Hat Enterprise Linux 8.x: Installation and Configuration*

**Note:** A valid SAP “S-User” id is required to display/access above SAP Notes.

The examples we provide will be using Red Hat Enterprise Linux. However, there is a parallel and mostly identical deployment using SUSE Linux if that is your preference.

### IBM PowerVS configuration options

Table 6-1 shows the current options for running SAP NetWeaver on PowerVS instances.

*Table 6-1 Current configuration options for SAP NetWeaver on PowerVS*

| Power System | SAPS per CPU Core | SAPS per CPU Thread (SMT-8) |
|---|---|---|
| S922 | 5,570 | 696.25 |
| E980 | 6,000 | 750 |

As additional configuration options might be available in the future, it would be a good idea to check the latest information in the IBM Cloud documentation.

Custom sizes for the IBM PowerVS instances are supported for use with SAP NetWeaver or S/4HANA in accordance with existing SAP recommendations. Sizing for your SAP NetWeaver or S/4HANA instances should follow the standard SAP sizing guidelines provided by using the SAPS benchmark calculations. For more information, *SAP Note 2855850*.

All SAP NetWeaver Application Server ABAP-based products and SAP NetWeaver Application Server Java-based products are supported on IBM Power Virtual Servers. As some of the older versions are no longer supported, refer to respective SAP Notes to validate and deploy supported combinations of these components on these SAP landscapes.

To set up RHEL 8.x Linux operating systems suitable for SAP NetWeaver or S/4HANA application landscapes, see *SAP Note 2772999 - Red Hat Enterprise Linux 8.x: Installation and Configuration.*

## 6.2.1 HA Solution for NetWeaver or S/4 Based on ABAP Platform 1709 or older

The Standalone Enqueue Server (ENSA1) is an older ASCS implementation. When using ENSA1 the Enqueue Client is required to fail over to the cluster node where the active ERS instance is running limiting your configuration flexibility. Ensure that the ERS instance is shut down and moved to the node where the ASCS instance was running when it was online because the ASCS instance has to access the shared memory where the ERS instance maintains a copy of the enqueue lock table to continue to keep track of the enqueue locks of active transactions.

When using stand alone enqueue server (ENSA1), the ASCS instance must always move to the node where the ERS instance is running and can only be deployed as a two node Pacemaker cluster. Detailed configuration guides for Red Hat can be found at Configuring SAP NetWeaver ASCS/ERS ENSA1 with Standalone Resources in RHEL 7.5+. The cluster configuration is illustrated in Figure 6-1 on page 86.

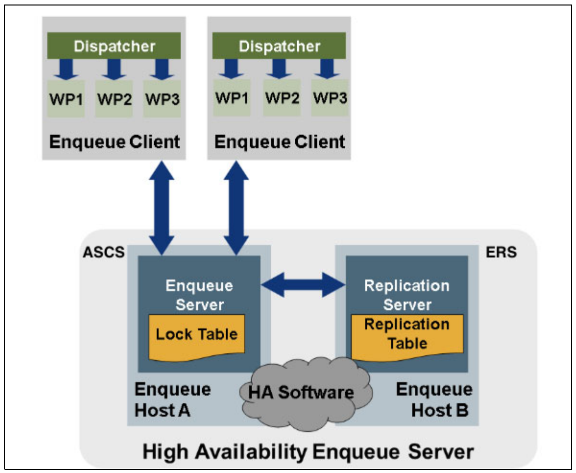


*Figure 6-1 Standalone Enqueue Server 1 (ENSA1)*[1]

This deployment architecture be upgraded to ENSA2 which is the current standard default deployment configuration and an improvement over the older setups. See 6.2.2, "HA Solution for S/4HANA based on ABAP Platform 1809 or newer" on page 86 for further details.

### 6.2.2 HA Solution for S/4HANA based on ABAP Platform 1809 or newer

The Standalone Enqueue Server has evolved into generation 2 with ABAP Platform 1809 and later, known as Standalone Enqueue Server 2, or ENSA2. In ENSA2, if the ASCS fails, it can start on a separate node in a cluster and the new instance will copy the lock entries from the enqueue replicator 2.

Figure 6-2 provides an illustration of the Standalone Enqueue Server which is a component of Application Server ABAP. ENSA2 provides a mechanism to ensure the HA of the lock table and its entries.

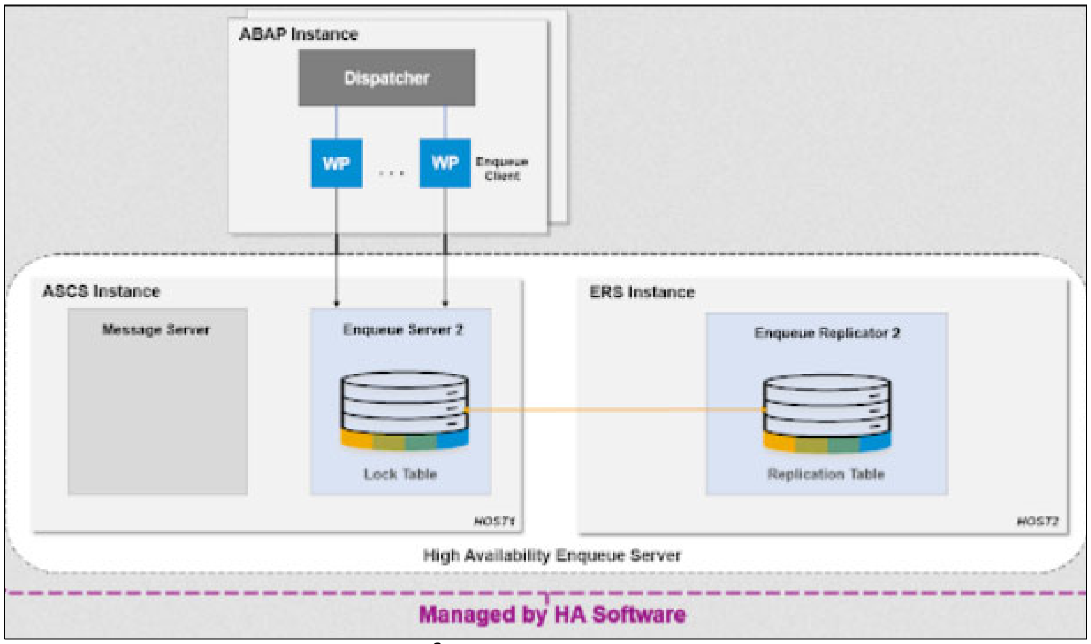


*Figure 6-2 ENSA2 Enqueue Replicator*[2]

[1] Red Hat HA Solutions for SAP

Starting with ABAP Application 1809, ENSA2 based deployment has become the default installation configuration for new installations.

There are two main supported scenarios as shown in Table 6-2.

*Table 6-2 Cluster configuration options*

| Supported Scenario | Description |
|---|---|
| Multi-Node Cluster | Multi-node clusters are supported since there is no need for ENSA2 ASCS to “follow” ERS |
| Two-node cluster | The existing or upgrades from ENSA1 to ENSA2 deployments are easily adjustable and usable with two-node clusters when switching from ENSA1 to ENSA2 |

Brand new installations can take advantage of the new features when designing the architecture, now having the option of choosing between a two-node cluster or a multi-node cluster. Figure 6-3 shows a reference architecture diagram of a typical 3-node cluster.

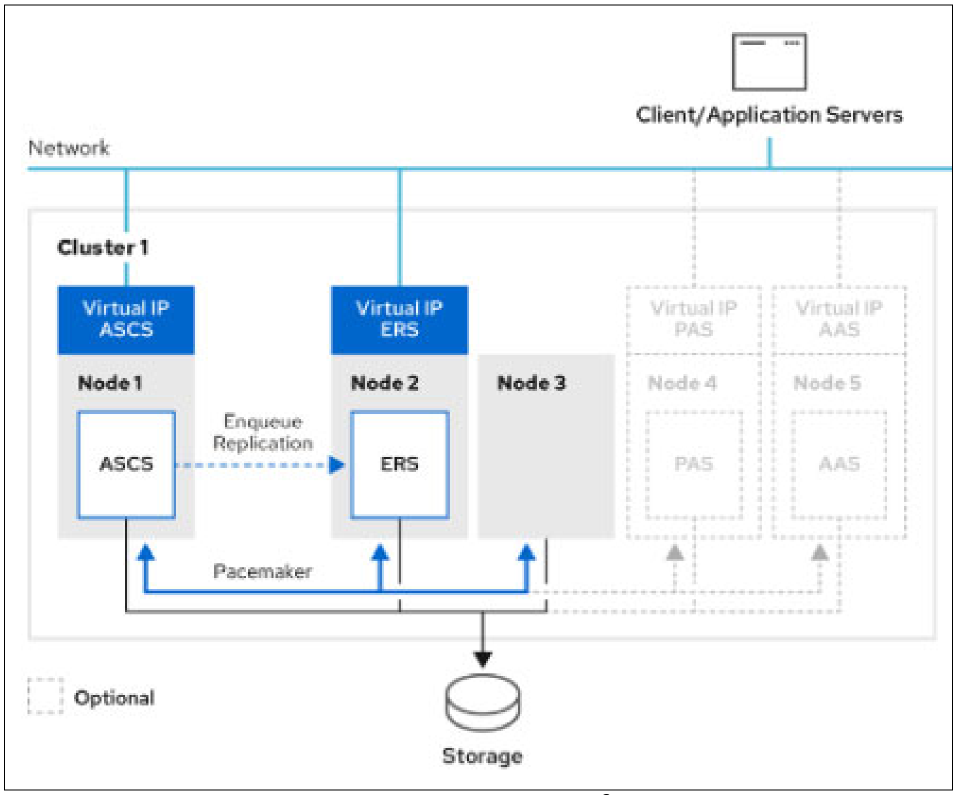


*Figure 6-3 ENSA2 Deployment Reference Architecture*[3]

**Note:** More nodes can be added to this design, depending upon the specific data center requirements.

[2] Red Hat documentation on HA solutions.
[3] Red Hat HA Solutions for SAP

## 6.2.3 Configuring HA for SAP S/4HANA (ASCS and ERS) in a RHEL HA Add-On cluster

The Red Hat documentation Red Hat HA Solutions for SAP HANA, S/4HANA, and NetWeaver based SAP Applications discusses various configuration aspects for SAP solutions in a RHEL HA Add-On cluster utilizing Pacemaker. The following information describes the configuration of ABAP SAP Central Services (ASCS) and Enqueue Replication Service (ERS) instances in a RHEL HA Add-On cluster on IBM Power Systems Virtual Server. The focus of this example configuration is on the second generation of the Standalone Enqueue Server, or ENSA2.

As mentioned before, starting with the release of SAP S/4HANA 1809, ENSA2 is installed by default and can be configured in either a two-node cluster or a multi-node cluster. This example uses the ENSA2 setup for a two-node RHEL HA Add-On cluster.

If the ASCS service fails in a two-node cluster, it restarts on the node where ERS is running. The lock entries for the SAP application are protected when they are rebuilt from the copy of the lock table that's available in the ERS. When the failed cluster node reactivates, the ERS instance moves to the other node (anti-collocation) to protect the lock table copy.

**Note:** This cluster is designed to protect the ASCS and ERS services. Install the SAP database instance and other SAP application servers on virtual server instances in a separate cluster.

### Installing on PowerVS

The following information outlines the configuration of ASCS and ERS in a RHEL HA Add-On cluster with Red Hat Enterprise Linux 8.4 (RHEL) by using IBM Power Systems Virtual Server instances as cluster nodes.

For more information, see the following Red Hat knowledge base articles:

- Configuring SAP S/4HANA ASCS/ERS with Standalone Enqueue Server 2 (ENSA2) in Pacemaker
- Support Policies for RHEL High Availability Clusters - Management of SAP S/4HANA in a cluster

### Prerequisites

Before starting, ensure that you understand and have satisfied the following prerequisites:

- A Red Hat Customer Portal account.
- An IBM Cloud account.
- A valid RHEL for SAP Applications or RHEL for SAP Solutions subscription is required to enable the repositories that you need to install SAP HANA and the resource agents for HA configurations.
- Virtual server instances need to fulfill the hardware and resource requirements for the SAP instances in scope.
- The setup described here uses shareable storage volumes accessible to both cluster nodes. Certain file systems are created on shareable storage volumes so that they can be mounted on both cluster nodes. These instance directories are shared:
  - `/usr/sap/<SID>/ASCS<Inst#>` of the ASCS instance.
  - `/usr/sap/<SID>/ERS<Inst#>` of the ERS instance.

  Make sure that the storage volumes that were created for those file systems are attached to both virtual server instances. During SAP instance installation and RHEL HA Add-On cluster configuration, each instance directory must be mounted on its appropriate node.

HA-LVM makes sure that each of the two instance directories is mounted on only one node at a time.

**Important:** Various storage configurations for instance directories, such as NFS mounts, are available. Opting for a different storage setup will necessitate adjustments in the setup procedures and the creation of appropriate cluster file system resources.

- The virtual host name for ASCS instance and ERS instance must meet the requirements as documented in Hostname of SAP ABAP Platform servers. Make sure that the virtual IP addresses for the SAP instances are assigned to a network adapter and that they can communicate in the network.
- Make sure that you subscribed to either RHEL for SAP Applications or RHEL for SAP Solutions and that you enabled the Red Hat Enterprise Linux for SAP Solutions for Power LE - Update Services for SAP Solutions 8.4 ppc64le (`rhel-8-for-ppc64le-sap-solutions-e4s-rpms`) repository. Set the release to 8.4 through the subscription manager.
- SAP application server instances require a common shared file system SAPMNT */sapmnt/<SID>* with read and write access, and other shared file systems such as `SAPTRANS /usr/sap/trans`. These file systems are typically provided by an external NFS server. The NFS server must be installed on virtual servers that are not part of the ENSA2 cluster.
- Configuring an active-passive NFS server in a Red Hat High Availability cluster describes the implementation of an active-passive NFS server in a RHEL HA Add-On cluster with Red Hat Enterprise Linux 8 by using virtual server instances in Power Systems Virtual Server.
- Make sure that all SAP installation media is available.

## Preparing nodes for SAP installation

This section describes how to prepare the nodes for an SAP installation.

### *Preparing environment variables*

To simplify the setup, prepare the following environment variables for user root on both cluster nodes. These environment variables are used in subsequent commands in the remainder of the instructions.

1. On both nodes, create a file with the environment variables shown in Example 6-1 making sure to adapt them to your configuration.

*Example 6-1 File “sap_env.sh” to be created on both nodes*

```
export SID=<SID>                    # SAP System ID (uppercase)
export sid=<sid>                    # SAP System ID (lowercase)

export ASCS_nr=<INSTNO>             # Instance Number for ASCS
export ERS_nr=<INSTNO>              # Instance Number for ERS

# virtual hostnames
export ASCS_vh=<virtual hostname>  # virtual hostname for ASCS
export ERS_vh=<virtual hostname>   # virtual hostname for ERS
export ASCS_ip=<IP address>        # virtual IP address for ASCS
export ERS_ip=<IP address>         # virtual IP address for ERS

# LVM storage for instance file systems
export ASCS_vg=<vg name>           # volume group name for ASCS
export ERS_vg=<vg name>            # volume group name for ERS

export ASCS_lv=<lv name>           # logical volume name for ASCS
export ERS_lv=<lv name>           # logical volume name for ERS
```

**Important:** It is recommended to use meaningful names for the volume groups and logical volumes that designate their content. For example, include the SID and ascs or ers in the name. Do not use hyphens in the volume group or logical volume names. For example:

- `s01ascsvg` and `s01ascslv`
- `s01ersvg` and `s01erslv`

2. Apply the variables in both nodes by running the command shown in Example 6-2.

*Example 6-2 Source "sap_env.sh"and set environment variables on both nodes.*

```
source sap_envs.sh
```

### *Assigning virtual IP addresses*

3. Check whether the virtual IP address for the SAP instance is present. Otherwise, you need to identify the correct network adapter to assign the IP address. The command is shown in Example 6-3.

*Example 6-3 On both nodes, check the list of currently active IP addresses*

```
ip -o -f inet address show | '/scope global/ {print $2, $4}'
```

Sample output from the previous command is shown in Example 6-4.

*Example 6-4 Sample output of IP addresses*

```
# ip -o -f inet address show | awk '/scope global/ {print $2, $4}'
env2 10.51.0.66/24
env3 10.52.0.41/24
env4 10.111.1.28/24
```

The device name of the network adapter appears in the first column. The second column lists the active IP addresses and the number of bits that are reserved for the netmask, which are separated by a slash.

4. If the virtual IP address for the SAP instance is not present, make sure that it is not erroneously set on another virtual server instance by running the command in Example 6-5.

*Example 6-5 Ping to ensure VIP not assigned*

```
ping -c 3 ${ASCS_vh}
```

The results of the ping should show that the address is unreachable as shown in Example 6-6.

*Example 6-6 Ping Sample Output*

```
# ping -c 2 cl-sap-scs
PING cl-sap-scs (10.111.1.248) 56(84) bytes of data.
From cl-sap-1.tst.ibm.com (10.111.1.28) icmp_seq=1 Destination Host Unreachable
From cl-sap-1.tst.ibm.com (10.111.1.28) icmp_seq=2 Destination Host Unreachable

--- cl-sap-ers ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 2112ms
pipe 3
```

**Note:** If the ping output shows Destination Host Unreachable, the IP address is available, and you can assign the IP alias to the virtual server instance. Use the correct device name of the network adapter that matches the subnet of the IP address.

5. Add the IP address definition on both of the nodes using the command in Example 6-7.

*Example 6-7 Add IP addresses on Nodes*

```
On Node1
# ip addr add ${ASCS_ip} dev env4
On Node2
# ip addr add ${ERS_ip} dev env4
```

**Note:** Adjust the device name for the network adapter based on your configuration.

## Preparing volume groups, logical volumes, and shared file systems

Shared storage is an important resource in an ENSA2 cluster. ASCS and ERS must be able to run on both nodes, and their runtime environment is stored in the shared storage volumes. All cluster nodes need to access the shared storage volumes, but only one node has exclusive read and write access to a volume.

### *Preparing HA Logical Volume Manager settings*

1. Edit file `/etc/lvm/lvm.conf` to include the system ID in the volume group as shown in Example 6-8.

*Example 6-8 On both nodes, edit the lvm.conffile*

```
On Both Nodes
# vi /etc/lvm/lvm.conf

Search for parameter system_id_source and change its value to "uname".
# system_id_source = "uname"
```

### *Identifying World Wide Names of shared storage volumes*

2. Identify the World Wide Name (WWN) for each storage volume that is in the shared volume groups.

- Log in to IBM Cloud to the Storage volumes view of Power Systems Virtual Server.
- Select your workspace.
- Filter for the volume prefix in the Storage volumes list, and identify all the World Wide Names of the volumes that are in scope for ASCS and ERS (the World Wide Name is a 32-digit hexadecimal number).

**Note:** Make sure that the attribute Shareable is On for those volumes.

3. In the Virtual server instances view, go to both virtual server instances of the cluster. Verify that all volumes that are in scope for ASCS and ERS are attached to both virtual server instances.

   If you need to attach a new storage volume to a virtual server instance, make sure that you rescan the SCSI bus to detect the new volume. Afterward, update the multi-path configuration of the virtual server instance. The command for this is shown in Example 6-9 on page 92, which should be run on both nodes.

*Example 6-9 Command to rescan for new devices and update multi-path*

```
# rescan-scsi-bus.sh && sleep 10 && multipathd reconfigure
```

4. Log in to both cluster nodes, and add the WWN to the environment variables of user root as shown in Example 6-10.

*Example 6-10 Add WWN to environment variables*

```
On NODE1, run the following command.
# export ASCS_pvid=<WWN>  # WWN of shared storage volume for ASCS

On NODE2, run the following command.
# export ERS_pvid=<WWN>   # WWN of shared storage volume for ERS
```

**Note:** The matching WWN values can also be found with the **pvs --all** command.

Make sure that you set the environment variable by using the hexadecimal number with lowercase letters.

### *Creating physical volumes*

5. Create physical volumes on the nodes for the attached storage volumes as shown in Example 6-11.

*Example 6-11 Create on both Nodes*

```
On NODE1
# pvcreate /dev/mapper/${ASCS_pvid}

On NODE2
# pvcreate /dev/mapper/${ERS_pvid}
```

Sample output from this command is shown in Example 6-12.

*Example 6-12 pvcreate output on each node*

```
# On NODE1
pvcreate /dev/mapper/${ERS_pvid}
Physical volume "/dev/mapper/3600507681081033570000000000002e31" successfully created.

# On NODE2
pvcreate /dev/mapper/${ASCS_pvid}
Physical volume "/dev/mapper/3600507681081033570000000000002ddc" successfully created.
```

### *Creating volume groups*

6. Create volume groups for ASCS and ERS using the commands shown in Example 6-13.

*Example 6-13 Create on both Nodes*

```
On NODE1
# vgcreate ${ASCS_vg} /dev/mapper/${ASCS_pvid}
Verify that the System ID is set.
# vgs -o+systemid

On NODE2
# vgcreate ${ERS_vg} /dev/mapper/${ERS_pvid}
Verify that the System ID is set.
# vgs -o+systemid
```

Sample output is shown in Example 6-14.

*Example 6-14 vgs output on each node*

```
# On NODE1
vgs -o+systemid
VG          #PV #LV #SN Attr   VSize   VFree   System ID
s01ascsvg     1   0   0 wz--n- <50.00g <50.00g cl-sap-1

# On NODE2
vgs -o+systemid
VG          #PV #LV #SN Attr   VSize   VFree   System ID
s01ersvg     1   0   0 wz--n- <50.00g <50.00g cl-sap-2
```

### *Creating logical volumes and file systems*

7. Create the logical volume for the ASCS and ERS, then format it as XFS file system as shown in Example 6-15.

*Example 6-15 Create Filesystems on both Nodes*

```
On NODE1
# lvcreate -l 100%FREE -n ${ASCS_lv} ${ASCS_vg}
# mkfs.xfs /dev/${ASCS_vg}/${ASCS_lv}

On NODE2
# lvcreate -l 100%FREE -n ${ERS_lv} ${ERS_vg}
# mkfs.xfs /dev/${ERS_vg}/${ERS_lv}
```

### *Making sure that a volume group is not activated on multiple cluster nodes*

8. Volume groups that are managed by the cluster must not activate automatically on startup. On both nodes, edit file `/etc/lvm/lvm.conf` and modify the `auto_activation_volume_list` entry to limit auto activation to specific volume groups as shown in Example 6-16.

*Example 6-16 Edit /etc/lvm/lvm.conf on both Nodes, recreate boot image and reboot*

```
On Both Nodes
# vi /etc/lvm/lvm.conf

Search for parameter auto_activation_volume_list and add the
volume groups, other than the volume group that you defined for
the NFS cluster, as entries in that list.
# auto_activation_volume_list = [ "rhel_root" ]

On both NODES, Rebuild the initramfs boot image to make sure that
the boot image does not activate a volume group that is controlled
by the cluster.
# dracut -H -f /boot/initramfs-$(uname -r).img $(uname -r)

Reboot both nodes.
# shutdown -t now -r
```

**Note:** For RHEL 8.5 and later, you can disable auto activation for a volume group when you create the volume group by specifying the **--setautoactivation n** flag for the **`vgcreate`** command.

### *Mounting the file systems for SAP installation*

9. Activate the volume groups and mount the SAP instance file systems. Run the commands shown in Example 6-17,

*Example 6-17 Run on Node1(ASCS) and Node 2(ERS)*

```
On NODE1 (ASCS)
# vgchange -a y ${ASCS_vg}
# mkdir -p /usr/sap/${SID}/ASCS${ASCS_nr}
# mount /dev/${ASCS_vg}/${ASCS_lv} /usr/sap/${SID}/ASCS${ASCS_nr}

On NODE2 (ERS)
# vgchange -a y ${ERS_vg}
# mkdir -p /usr/sap/${SID}/ERS${ERS_nr}
# mount /dev/${ERS_vg}/${ERS_lv} /usr/sap/${SID}/ERS${ERS_nr}
```

### *Mounting the required NFS file systems*

10. On both nodes, make sure that the NFS file systems `/sapmnt` and `/usr/sap/trans` are mounted by running the commands shown in Example 6-18.

*Example 6-18 Check NFS mounts*

```
On NODE1 (ASCS)
# mount | grep nfs
On NODE2 (ERS)
# mount | grep nfs
```

## Installing SAP instances

Use the SAP Software Provisioning Manager (SWPM) to install all instances.

Next it is time to install SAP instances on the cluster nodes. This is done using the SAP installer SWPM.

- Install ASCS instance on NODE1 by using the virtual hostname ${ASCS_vh} that is associated with the virtual IP address for ASCS:
  **`<swpm>/sapinst SAPINST_USE_HOSTNAME=${ASCS_vh}`**
- Install ERS instance on NODE2 by using the virtual hostname ${ERS_vh} that is associated with the virtual IP address for ERS:
  **`<swpm>/sapinst SAPINST_USE_HOSTNAME=${ERS_vh}`**
- Install instances outside the cluster:
  - DB instance
  - PAS instance
  - AAS instances

## Installing and setting up the RHEL HA Add-On cluster

Now that your SAP landscape is installed and ready to go it is time to enable the Pacemaker function to manage the high availability functions. For more information, see Implementing a RHEL HA Add-On cluster on IBM Power Virtual Server.

## Preparing ASCS and ERS instances for the cluster integration

Use the following steps to prepare the SAP instances for the cluster integration.

1. Adjust the SAP services file `/usr/sap/sapservices` to prevent automatic start of the sapstartsrv instance agent for both ASCS and ERS instance after a reboot as shown in Example 6-19 on page 95.

*Example 6-19 On both nodes comment the sapstartsrv entries for ASCS and ERS*

```
On NODE1 (ASCS)
# vi /usr/sap/sapservices

On NODE2 (ERS)
# vi /usr/sap/sapservices
```

Example 6-20 shows an example of what line to comment out.

*Example 6-20 Example Commented sapstartsrv entries*

```
On NODE1 and NODE2
#LD_LIBRARY_PATH=/usr/sap/S01/ASCS01/exe:$LD_LIBRARY_PATH;export
LD_LIBRARY_PATH;/usr/sap/S01/ASCS01/exe/sapstartsrv
pf=/usr/sap/S01/SYS/profile/S01_ASCS01_cl-sap-scs -D -u s01adm
```

2. Create the mount points for the instance file systems and adjust their ownership using the commands shown in Example 6-21.

*Example 6-21 Create mount points for ASCS and ERS*

```
On NODE1 (ASCS)
# mkdir /usr/sap/${SID}/ASCS${ASCS_nr}
# chown ${sid}adm:sapsys /usr/sap/${SID}/ASCS${ASCS_nr}

On NODE2 (ERS)
# mkdir /usr/sap/${SID}/ERS${ERS_nr}
# chown ${sid}adm:sapsys /usr/sap/${SID}/ERS${ERS_nr}
```

3. When the SAP ASCS instance is installed on a Power Systems Virtual Server instance, the SAP license mechanism relies on the partition UUID. For more information, see *SAP note 2879336 - Hardware key based on unique ID*.

   On both nodes, run the commands shown in Example 6-22 as user *<sid>adm* to identify the **HARDWARE KEY** of the node.

*Example 6-22 On both Nodes Get SAP License hardware Key*

```
On Both NODEs
# sudo -i -u ${sid}adm -- sh -c 'saplikey -get'
```

Sample output is provided in Example 6-23.

*Example 6-23 Sample Hardware Key Output*

```
$ sudo -i -u ${sid}adm -- sh -c 'saplikey -get'

saplikey: HARDWARE KEY = H1428224519
```

Note the HARDWARE KEY of each node as you need both hardware keys to request two different SAP license keys. Check the following SAP notes for more information about requesting SAP license keys:

- *SAP Note - 2879336 - Hardware key based on unique ID*
- *SAP Note - 2662880 - How to request SAP license keys for failover systems*

4. Install the required software packages. The **resource-agents-sap** includes the SAP Instance cluster resource agent for managing the SAP instances.

Unless **sap_cluster_connector** is configured for the SAP instance, the RHEL HA Add-On cluster considers any state change of the instance as an issue. If other SAP tools such as **sapcontrol** are used to manage the instance, then **sap_cluster_connector** grants permission to control SAP instances that are running inside the cluster. If the SAP instances are managed by only cluster tools, the implementation of **sap_cluster_connector** is not necessary.

Install the packages for the resource agent and the SAP Cluster Connector library. For more information, see How to enable the SAP HA Interface for SAP ABAP application server instances managed by the RHEL HA Add-On

On both nodes, run the command shown in Example 6-24. If needed, use **subscription-manager** to enable the SAP NetWeaver repository.

*Example 6-24 Enable Red Hat subscriptions*

```
On Both NODEs
# subscription-manager repos --enable="rhel-8-for-ppc64le-sap-netweaver-e4s-rpms"
```

5. Add user ${sid}admto the haclient group using the command in Example 6-25.

*Example 6-25 Add user to haclient group*

```
On Both NODEs
# usermod -a -G haclient ${sid}adm
```

6. Modify the start profiles of all SAP instances that are managed by SAP tools outside the cluster. Both ASCS and ERS instances can be controlled by the RHEL HA Add-On cluster and its resource agents. Adjust the SAP instance profiles to prevent an automatic restart of instance processes using the commands shown in Example 6-26.

*Example 6-26 Update SAP Profiles for ASCS and ERS*

```
On NODE1
Change all occurrences of Restart_Program to Start_Program in the instance profile of
both ASCS and ERS.

# cd /sapmnt/${SID}/profile
# sed -i -e 's/Restart_Program_\([0-9][0-9]\)/Start_Program_\1/'
${SID}_ASCS${ASCS_nr}_${ASCS_vh}
# sed -i -e 's/Restart_Program_\([0-9][0-9]\)/Start_Program_\1/'
${SID}_ERS${ERS_nr}_${ERS_vh}

Add the following two lines at the end of the SAP instance profile to configure
sap_cluster_connector for the ASCS and ERS instance, and activate the integration.
   service/halib = $(DIR_EXECUTABLE)/saphascriptco.so
   service/halib_cluster_connector = /usr/bin/sap_cluster_connector
```

## Configuring ASCS and ERS cluster resources

Up to this point, the following are assumed:

- A RHEL HA Add-On cluster is running on both virtual server instances and fencing of the nodes was tested.
- The SAP System is running.
  - SAP ASCS is installed and active on node 1 of the cluster.
  - SAP ERS is installed and active on node 2 of the cluster.
- All steps in Prepare ASCS and ERS instances for the cluster integration are complete.

1. Create a cloned Filesystem cluster resource to mount the SAPMNT share from an external NFS server to all cluster nodes. Make sure that the environment variable ${NFS_vh} is set to the virtual hostname of your NFS server ${NFS_vh}, and ${NFS_options} according to your mount options. Use the commands shown in Example 6-27.

*Example 6-27 Set Mount Options*

```
On Both NODEs
# export NFS_options="rw,sec=sys"

On NODE1
# pcs resource create fs_sapmnt Filesystem \
     device="${NFS_vh}:/${SID}" \
     directory="/sapmnt/${SID}" \
     fstype='nfs' \
     "options=${NFS_options}" \
     clone interleave=true
```

### *Configuring the ASCS resource group*

2. Create a resource for the virtual IP address of the ASCS as shown in Example 6-28.

*Example 6-28 Create ASCS Resource Group*

```
On NODE1
# pcs resource create ${sid}_vip_ascs${ASCS_nr} IPaddr2 \
  ip=${ASCS_ip} \
  --group ${sid}_ascs${ASCS_nr}_group
```

In Example 6-29 we show creating resources for an HA-LVM file system on a shared storage volume, we create resources for LVM-activate and for the instance file system of the ASCS.

*Example 6-29 Create Resource with LVM Filesystem*

```
On NODE1
# pcs resource create ${sid}_fs_ascs${ASCS_nr}_lvm LVM-activate \
     vgname="${ASCS_vg}" \
     vg_access_mode=system_id \
     --group ${sid}_ascs${ASCS_nr}_group

# pcs resource create ${sid}_fs_ascs${ASCS_nr} Filesystem \
     device="/dev/mapper/${ASCS_vg}-${ASCS_lv}" \
     directory=/usr/sap/${SID}/ASCS${ASCS_nr} \
     fstype=xfs \
     --group ${sid}_ascs${ASCS_nr}_group
```

In the alternative example shown in Example 6-30, the instance file system of the ASCS is provided by an HA NFS server, only the file system resource is required. Make sure that you defined the environment variable **${NFS_vh}** according to your NFS server, and that you created a directory **${SID}/ASCS** under the NFS root directory during the SAP installation of the ASCS instance.

*Example 6-30 Create Resource with NFS Filesystem*

```
On NODE1
# pcs resource create ${sid}_fs_ascs${ASCS_nr} Filesystem \
     device="${NFS_vh}:${SID}/ASCS" \
     directory=/usr/sap/${SID}/ASCS${ASCS_nr} \
     fstype=nfs \
     force_unmount=safe \
     op start interval=0 timeout=60 \
     op stop interval=0 timeout=120 \
     --group ${sid}_ascs${ASCS_nr}_group
```

3. Create a resource for managing the ASCS instance as shown in Example 6-31.

*Example 6-31 Create Resource for Managing ASCS Instance*

```
On NODE1
# pcs resource create ${sid}_ascs${ASCS_nr} SAPInstance \
     InstanceName="${SID}_ASCS${ASCS_nr}_${ASCS_vh}" \
     START_PROFILE=/sapmnt/${SID}/profile/${SID}_ASCS${ASCS_nr}_${ASCS_vh} \
     AUTOMATIC_RECOVER=false \
     meta resource-stickiness=5000 \
     migration-threshold=1 failure-timeout=60 \
     op monitor interval=20 on-fail=restart timeout=60 \
     op start interval=0 timeout=600 \
     op stop interval=0 timeout=600 \
     --group ${sid}_ascs${ASCS_nr}_group
```

The meta **resource-stickiness=5000** option is used to balance the failover constraint with ERS so that the resource stays on the node where it started and does not migrate uncontrollably in the cluster.

4. Add a resource stickiness to the group to make sure that the ASCS remains on the node as shown in Example 6-32.

*Example 6-32 Add a resource with stickiness to remain on node*

```
On NODE1
# pcs resource meta ${sid}_ascs${ASCS_nr}_group     resource-stickiness=3000
```

### *Configuring ERS resource group*

5. Create a resource for the virtual IP address of the ERS as shown in Example 6-33.

*Example 6-33 Create ERS Resource Group*

```
On NODE1
# pcs resource create ${sid}_vip_ers${ERS_nr} IPaddr2 \
     ip=${ERS_ip} \
     --group ${sid}_ers${ERS_nr}_group
```

Example 6-34 shows creating resources for an HA-LVM file system on a shared storage volume, you create resources for LVM-activate and for the instance file system of the ERS.

*Example 6-34 Create Resource with LVM Filesystem*

```
On NODE1
# pcs resource create ${sid}_fs_ers${ERS_nr}_lvm LVM-activate \
     vgname="${ERS_vg}" \
     vg_access_mode=system_id \
     --group ${sid}_ers${ERS_nr}_group
# pcs resource create ${sid}_fs_ers${ERS_nr} Filesystem \
     device="/dev/mapper/${ERS_vg}-${ERS_lv}" \
     directory=/usr/sap/${SID}/ERS${ERS_nr} \
     fstype=xfs \
     --group ${sid}_ers${ERS_nr}_group
```

In the alternative example shown in Example 6-35, the instance file system of the ERS is provided by an HA NFS server, only the file system resource is required. Make sure that you defined the environment variable **${NFS_vh}** according to your NFS server, and that you created a directory **${SID}/ERS** under the NFS root directory during the SAP installation of the ERS instance.

*Example 6-35 Create Resource with NFS Filesystem*

```
On NODE1

# pcs resource create ${sid}_fs_ers${ERS_nr} Filesystem \
     device="${NFS_vh}:${SID}/ERS" \
     directory=/usr/sap/${SID}/ERS${ERS_nr} \
     fstype=nfs \
     force_unmount=safe \
     op start interval=0 timeout=60 \
     op stop interval=0 timeout=120 \
     --group ${sid}_ers${ERS_nr}_group
```

6. Create a resource for managing the ERS instance which is shown in Example 6-36.

*Example 6-36 Create Resource for Managing ERS Instance*

```
On NODE1
# pcs resource create ${sid}_ers${ERS_nr} SAPInstance \
     InstanceName="${SID}_ERS${ERS_nr}_${ERS_vh}" \
     START_PROFILE=/sapmnt/${SID}/profile/${SID}_ERS${ERS_nr}_${ERS_vh} \
     AUTOMATIC_RECOVER=false \
     IS_ERS=true \
     op monitor interval=20 on-fail=restart timeout=60 \
     op start interval=0 timeout=600 \
     op stop interval=0 timeout=600 \
     --group ${sid}_ers${ERS_nr}_group
```

### *Configuring constraints*

7. A collocation constraint prevents resource groups **${sid}_ascs${ASCS_nr}_group** and **${sid}_ers${ERS_nr}_group** from being active on the same node whenever possible. The stickiness score of **-5000** makes sure that they run on the same node if only a single node is available. See Example 6-37 for the details.

*Example 6-37 Add collocation constraints and Order constraints*

```
Add constraint
# pcs constraint colocation add \
     ${sid}_ers${ERS_nr}_group with ${sid}_ascs${ASCS_nr}_group -5000

Add the following Order constraint to control resource group ${sid}_ascs${ASCS_nr}_group
starts before ${sid}_ers${ERS_nr}_group.
# pcs constraint order start \
  ${sid}_ascs${ASCS_nr}_group then stop ${sid}_ers${ERS_nr}_group \
  symmetrical=false \
  kind=Optional

Add following two order constraints make sure that the file system SAPMNT mounts before
resource groups ${sid}_ascs${ASCS_nr}_group and ${sid}_ers${ERS_nr}_group start.
```

```
# pcs constraint order fs_sapmnt-clone then ${sid}_ascs${ASCS_nr}_group
# pcs constraint order fs_sapmnt-clone then ${sid}_ers${ERS_nr}_group
```

At this point, cluster setup is complete.

## Testing SAP ENSA2 clusters

It is vital to thoroughly test the cluster configuration to make sure that the cluster is working correctly. The following information provides a few sample failover test scenarios, but is not a complete list of test scenarios.

For example, the description of each test case includes the following information.

- Component under test
- Description of the test
- Prerequisites and the initial state before failover test
- Test procedure
- Expected behavior and results
- Recovery procedure

### *Test1 - Testing failure of the ASCS instance*

This test is testing the failover of the ASCS instance when the ASCS service fails.

- **Test1 - Description**

  Simulate a failure of the SAP ASCS instance that is running on NODE1.

- **Test1 - Prerequisites**

  - A functional two-node RHEL HA Add-On cluster for SAP ENSA2.
  - Both cluster nodes are active.
  - Cluster is started on NODE1 and NODE2.
    - Resource group ${sid}_ascs${ASCS_nr}_group is active on NODE1.
    - Resources ${sid}_vip_ascs${ASCS_nr}, ${sid}_fs_ascs${ASCS_nr}_lvm, ${sid}_fs_ascs${ASCS_nr} and ${sid}_ascs${ASCS_nr} are **Started** on NODE1.
    - Resource group ${sid}_ers${ERS_nr}_group is active on NODE2.
    - Resources ${sid}_vip_ers${ERS_nr}, ${sid}_fs_ers${ERS_nr}_lvm, ${sid}_fs_ers${ERS_nr} and ${sid}_ers${ERS_nr} are **Started** on NODE2.

  To validate the status of the SAP instances and show where each instance is running, run the command shown in Example 6-38. The expected starting point is:

    - ASCS instance is running on NODE1.
    - ERS instance is running on NODE2.

  *Example 6-38 Cluster Status*

  ```
  # pcs status
  ```

  A sample output is shown in Example 6-39.

  *Example 6-39 Sample Cluster Status*

  ```
  # pcs status
  Cluster name: SAP_ASCS
  Cluster Summary:
    * Stack: corosync
    * Current DC: cl-sap-1 (version 2.0.5-9.el8_4.5-ba59be7122) - partition with quorum
    * Last updated: Tue Feb 14 07:59:16 2023
  ```

```
  * Last change:  Tue Feb 14 05:02:22 2023 by hacluster via crmd on cl-sap-1
  * 2 nodes configured
  * 11 resource instances configured

Node List:
  * Online: [ cl-sap-1 cl-sap-2 ]

Full List of Resources:
  * res_fence_ibm_powervs(stonith:fence_ibm_powervs): Started cl-sap-2
  * Resource Group: s01_ascs01_group:
    * s01_vip_ascs01(ocf::heartbeat:IPaddr2): Started cl-sap-1
    * s01_fs_ascs01_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-1
    * s01_fs_ascs01(ocf::heartbeat:Filesystem): Started cl-sap-1
    * s01_ascs01(ocf::heartbeat:SAPInstance): Started cl-sap-1
  * Resource Group: s01_ers02_group:
    * s01_vip_ers02(ocf::heartbeat:IPaddr2): Started cl-sap-2
    * s01_fs_ers02_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-2
    * s01_fs_ers02(ocf::heartbeat:Filesystem): Started cl-sap-2
    * s01_ers02(ocf::heartbeat:SAPInstance): Started cl-sap-2
  * Clone Set: fs_sapmnt-clone [fs_sapmnt]:
    * Started: [ cl-sap-1 cl-sap-2 ]

Daemon Status:
  corosync: active/disabled
  Pacemaker: active/disabled
  pcsd: active/enabled
```

- **Test1 - Test Procedure**

  Forcefully terminate the SAP ASCS instance by sending a SIGKILL signal as user ${sid}adm to the enqueue server. On NODE1, identify the PID of the enqueue server using the command shown in Example 6-40.

  *Example 6-40 On Node1 identify PID for Enqueue Server*

  ```
  # pgrep -af "(en|enq).sap"
  ```

  The output of the command is shown in Example 6-41. After discovering the PID, use the **kill** command as shown in the example.

  *Example 6-41 Locating PID and issuing kill command*

  ```
  # pgrep -af "(en|enq).sap"
  30186 en.sapS01_ASCS01 pf=/usr/sap/S01/SYS/profile/S01_ASCS01_cl-sap-scs
  # kill -9 30186
  ```

- **Test1 - Expected behavior**

  The following actions should occur after you issue the **kill** command for the ASCS instance.

  – SAP ASCS instance on NODE1 fails.

  – The cluster detects the failure of the ASCS instance.

  – The cluster stops the dependent resources on NODE1 (virtual IP address, file system **/usr/sap/${SID}/ASCS${ASCS_nr}**, and the LVM resources), and acquires them on NODE2.

  – The cluster starts the ASCS on NODE2.

  – The cluster stops the ERS instance on NODE2.

– The cluster stops the dependent resources on NODE1 (virtual IP address, file system **/usr/sap/${SID}/ERS${ERS_nr}**, and the LVM resources), and acquires them on NODE1.

– The cluster starts the ERS on NODE1.

To validate the outcome, after a few seconds check the status using the command shown in Example 6-42.

*Example 6-42 Command to check status*

```
# pcs status
```

The output should look similar to what is shown in Example 6-43.

*Example 6-43 Status after failover*

```
# pcs status
Cluster name: SAP_ASCS
Cluster Summary:
  * Stack: corosync
  * Current DC: cl-sap-1 (version 2.0.5-9.el8_4.5-ba59be7122) - partition with quorum
  * Last updated: Tue Feb 14 08:10:18 2023
  * Last change:  Tue Feb 14 05:02:22 2023 by hacluster via crmd on cl-sap-1
  * 2 nodes configured
  * 11 resource instances configured

Node List:
  * Online: [ cl-sap-1 cl-sap-2 ]

Full List of Resources:
  * res_fence_ibm_powervs(stonith:fence_ibm_powervs): Started cl-sap-2
  * Resource Group: s01_ascs01_group:
    * s01_vip_ascs01(ocf::heartbeat:IPaddr2): Started cl-sap-2
    * s01_fs_ascs01_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-2
    * s01_fs_ascs01(ocf::heartbeat:Filesystem): Started cl-sap-2
    * s01_ascs01(ocf::heartbeat:SAPInstance): Started cl-sap-2
  * Resource Group: s01_ers02_group:
    * s01_vip_ers02(ocf::heartbeat:IPaddr2): Started cl-sap-1
    * s01_fs_ers02_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-1
    * s01_fs_ers02(ocf::heartbeat:Filesystem): Started cl-sap-1
    * s01_ers02(ocf::heartbeat:SAPInstance): Started cl-sap-1
  * Clone Set: fs_sapmnt-clone [fs_sapmnt]:
    * Started: [ cl-sap-1 cl-sap-2 ]

Daemon Status:
  corosync: active/disabled
  Pacemaker: active/disabled
  pcsd: active/enabled
```

### Test2 - Testing failure of the node that is running the ASCS instance

This test shows how Pacemaker responds to a node failure when the node is running the ASCS instance.

- **Test2 - Description**

  Simulate a failure of the node where the ASCS instance is running.

- **Test2 - Prerequisites**

  – A functional two-node RHEL HA Add-On cluster for SAP ENSA2.
  – Both cluster nodes are active.

- Cluster is started on NODE1 and NODE2.
  - Resource group ${sid}_ascs${ASCS_nr}_group is active on NODE2.
  - Resources ${sid}_vip_ascs${ASCS_nr}, ${sid}_fs_ascs${ASCS_nr}_lvm, ${sid}_fs_ascs${ASCS_nr} and ${sid}_ascs${ASCS_nr} are **Started** on NODE2.
  - Resource group ${sid}_ers${ERS_nr}_group is active on NODE1.
  - Resources ${sid}_vip_ers${ERS_nr}, ${sid}_fs_ers${ERS_nr}_lvm, ${sid}_fs_ers${ERS_nr} and ${sid}_ers${ERS_nr} are **Started** on NODE1.
- Check SAP instance processes:
  - ASCS instance is running on NODE2.
  - ERS instance is running on NODE1.

- **Test2 - Test procedure**

  Forcefully terminate NODE2 by sending a **fast-restart** system request as shown in Example 6-44.

*Example 6-44 Fast-restart on NODE2*

```
sync; echo b > /proc/sysrq-trigger
```

- **Test2 - Expected behavior**
  - NODE2 restarts.
  - The cluster detects the failed node and sets its state to offline (UNCLEAN).
  - The cluster acquires the ASCS resources (virtual IP address, file system /usr/sap/${SID}/ASCS${ASCS_nr}, and the LVM items) on NODE1.
  - The cluster starts the ASCS on NODE1.

**Note:** The second node is offline and both resource groups are running on the first node.

Check the status with the command in Example 6-45.

*Example 6-45 Check Cluster Status.*

```
# pcs status
```

Example 6-46 shows the expected output showing all services running on NODE1.

*Example 6-46 Output showing cluster status after Node2 failure*

```
# pcs status
Cluster name: SAP_ASCS
Cluster Summary:
  * Stack: corosync
  * Current DC: cl-sap-1 (version 2.0.5-9.el8_4.5-ba59be7122) - partition with quorum
  * Last updated: Tue Feb 14 08:34:16 2023
  * Last change:  Tue Feb 14 08:34:04 2023 by hacluster via crmd on cl-sap-1
  * 2 nodes configured
  * 11 resource instances configured

Node List:
  * Online: [ cl-sap-1 ]
  * OFFLINE: [ cl-sap-2 ]

Full List of Resources:
  * res_fence_ibm_powervs(stonith:fence_ibm_powervs): Started cl-sap-1
  * Resource Group: s01_ascs01_group:
    * s01_vip_ascs01(ocf::heartbeat:IPaddr2): Started cl-sap-1
    * s01_fs_ascs01_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-1
    * s01_fs_ascs01(ocf::heartbeat:Filesystem): Started cl-sap-1
    * s01_ascs01(ocf::heartbeat:SAPInstance): Started cl-sap-1
  * Resource Group: s01_ers02_group:
    * s01_vip_ers02(ocf::heartbeat:IPaddr2): Started cl-sap-1
    * s01_fs_ers02_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-1
    * s01_fs_ers02(ocf::heartbeat:Filesystem): Started cl-sap-1
    * s01_ers02(ocf::heartbeat:SAPInstance): Started cl-sap-1
  * Clone Set: fs_sapmnt-clone [fs_sapmnt]:
    * Started: [ cl-sap-1 ]
    * Stopped: [ cl-sap-2 ]

Daemon Status:
  corosync: active/disabled
  Pacemaker: active/disabled
  pcsd: active/enabled
```

- **Test2 - Recovery procedure**

  Wait until NODE2 restarts, then restart the cluster framework using the command shown in Example 6-47.

*Example 6-47 On Node1 Start Cluster.*

```
pcs cluster start
```

The results of the cluster start are:

- The cluster starts on NODE2 and acquires the ERS resources (virtual IP address, file system `/usr/sap/${SID}/ERS${ERS_nr}`, and the LVM resources) on NODE2.
- The cluster starts the ERS instance on NODE2.

Wait a moment and check the status with the command as shown in Example 6-48, which shows that the ERS resource group moved to the second node.

*Example 6-48 Cluster status after recovery*

```
# pcs status
Cluster name: SAP_ASCS
Cluster Summary:
  * Stack: corosync
  * Current DC: cl-sap-1 (version 2.0.5-9.el8_4.5-ba59be7122) - partition with quorum
  * Last updated: Tue Feb 14 08:41:23 2023
  * Last change:  Tue Feb 14 08:34:04 2023 by hacluster via crmd on cl-sap-1
  * 2 nodes configured
  * 11 resource instances configured

Node List:
  * Online: [ cl-sap-1 cl-sap-2 ]

Full List of Resources:
  * res_fence_ibm_powervs(stonith:fence_ibm_powervs): Started cl-sap-1
  * Resource Group: s01_ascs01_group:
    * s01_vip_ascs01(ocf::heartbeat:IPaddr2): Started cl-sap-1
    * s01_fs_ascs01_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-1
    * s01_fs_ascs01(ocf::heartbeat:Filesystem): Started cl-sap-1
    * s01_ascs01(ocf::heartbeat:SAPInstance): Started cl-sap-1
  * Resource Group: s01_ers02_group:
    * s01_vip_ers02(ocf::heartbeat:IPaddr2): Started cl-sap-2
    * s01_fs_ers02_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-2
    * s01_fs_ers02(ocf::heartbeat:Filesystem): Started cl-sap-2
    * s01_ers02(ocf::heartbeat:SAPInstance): Started cl-sap-2
  * Clone Set: fs_sapmnt-clone [fs_sapmnt]:
    * Started: [ cl-sap-1 cl-sap-2 ]
```

```
Daemon Status:
  corosync: active/disabled
  Pacemaker: active/disabled
  pcsd: active/enabled
```

### *Test3 - Testing failure of the ERS instance*

This test will show the results of the ERS service on NODE2.

- **Test3 - Description**

  Simulate a failure of the ERS instance while it is running.

- **Test3 - Prerequisites**

  – A functional two-node RHEL HA Add-On cluster for SAP ENSA2.
  – Both cluster nodes are active.
  – Cluster starts on NODE1 and NODE2.
    - Resource group ${sid}_ascs${ASCS_nr}_group is active on NODE1.
    - Resources ${sid}_vip_ascs${ASCS_nr}, ${sid}_fs_ascs${ASCS_nr}_lvm, ${sid}_fs_ascs${ASCS_nr} and ${sid}_ascs${ASCS_nr} are **Started** on NODE1.
    - Resource group ${sid}_ers${ERS_nr}_group is active on NODE2.
    - Resources ${sid}_vip_ers${ERS_nr}, ${sid}_fs_ers${ERS_nr}_lvm, ${sid}_fs_ers${ERS_nr} and ${sid}_ers${ERS_nr} are **Started** on NODE2.

  – Check SAP instance processes:
    - ASCS instance is running on NODE1.
    - ERS instance is running on NODE2.

- Test3 - Test Procedure

  Forcefully terminate the SAP ERS instance by sending a SIGKILL signal as shown in Example 6-49.

*Example 6-49 Identify the PID of the enqueue replication server and send kill command*

```
pgrep -af "(er|enqr).sap"
2527198 er.sapS01_ERS02 pf=/usr/sap/S01/ERS02/profile/S01_ERS02_cl-sap-ers NR=01
kill -9 2527198
```

- **Test3 - Expected behavior**

  The following actions should occur as a result of shutting down the ERS service.

  – SAP Enqueue Replication Server on NODE2 fails immediately.
  – The cluster detects the stopped ERS and marks the resource as failed.
  – The cluster restarts the ERS on NODE2.

  Example 6-50 shows running the **`pcs status`** command and the expected results. The **${sid}_ers${ERS_nr}** ERS resource restarted on the second node. If you run the **pcs status** command too soon, you might see the ERS resource briefly in status **FAILED**. See Example 6-50 for sample results.

*Example 6-50 Cluster status after ERS service failure*

```
# pcs status
Cluster name: SAP_ASCS
Cluster Summary:
  * Stack: corosync
  * Current DC: cl-sap-1 (version 2.0.5-9.el8_4.5-ba59be7122) - partition with quorum
  * Last updated: Tue Feb 14 08:50:53 2023
  * Last change:  Tue Feb 14 08:50:50 2023 by hacluster via crmd on cl-sap-2
  * 2 nodes configured
  * 11 resource instances configured

Node List:
  * Online: [ cl-sap-1 cl-sap-2 ]

Full List of Resources:
  * res_fence_ibm_powervs(stonith:fence_ibm_powervs): Started cl-sap-1
  * Resource Group: s01_ascs01_group:
    * s01_vip_ascs01(ocf::heartbeat:IPaddr2): Started cl-sap-1
    * s01_fs_ascs01_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-1
    * s01_fs_ascs01(ocf::heartbeat:Filesystem): Started cl-sap-1
    * s01_ascs01(ocf::heartbeat:SAPInstance): Started cl-sap-1
  * Resource Group: s01_ers02_group:
    * s01_vip_ers02(ocf::heartbeat:IPaddr2): Started cl-sap-2
    * s01_fs_ers02_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-2
    * s01_fs_ers02(ocf::heartbeat:Filesystem): Started cl-sap-2
    * s01_ers02(ocf::heartbeat:SAPInstance): Started cl-sap-2
  * Clone Set: fs_sapmnt-clone [fs_sapmnt]:
    * Started: [ cl-sap-1 cl-sap-2 ]

Daemon Status:
  corosync: active/disabled
  Pacemaker: active/disabled
  pcsd: active/enabled
```

#### Test3 - Recovery Procedure

Run the commands in Example 6-51 to refresh the cluster.

*Example 6-51 On NODE2 Run following commands*

```
pcs resource refresh
pcs status --full
```

### *Test4 - Testing the manual move of the ASCS instance*

This test moves the ASCS instance to NODE2 using SAP control interfaces as in a maintenance action.

- **Test4 - Description**

Use SAP Control commands to move the ASCS instance to the other node for maintenance purposes.

- **Test4 - Prerequisites**
  - A functional two-node RHEL HA Add-On cluster for SAP ENSA2.
  - The **sap_cluster_connector** is installed and configured.
  - Both cluster nodes are active.
  - Cluster is started on NODE1 and NODE2.
    - Resource group ${sid}_ascs${ASCS_nr}_group is active on NODE1.
    - Resources ${sid}_vip_ascs${ASCS_nr}, ${sid}_fs_ascs${ASCS_nr}_lvm, ${sid}_fs_ascs${ASCS_nr} and ${sid}_ascs${ASCS_nr} are **Started** on NODE1.
    - Resource group ${sid}_ers${ERS_nr}_group is active on NODE2.
    - Resources ${sid}_vip_ers${ERS_nr}, ${sid}_fs_ers${ERS_nr}_lvm, ${sid}_fs_ers${ERS_nr} and ${sid}_ers${ERS_nr} are **Started** on NODE2.
  - Check SAP instance processes:

- ASCS instance is running on NODE1.
- ERS instance is running on NODE2.

**Test4 - Test Procedure**

Log in to NODE1 and run **`sapcontrol`** to move the ASCS instance to the other node. The command is shown in Example 6-52.

*Example 6-52 On NODE1 Run following commands*

```
sudo -i -u ${sid}adm -- sh -c "sapcontrol -nr ${ASCS_nr} -function HAFailoverToNode"
```

- **Test4 - Expected behavior**

The **`sapcontrol`** function interacts with the cluster through the sap-cluster-connector. The cluster creates location constraints to move the resource. Check the status as shown in Example 6-53.

*Example 6-53 Status after move of ASCS service*

```
# pcs status
Cluster name: SAP_ASCS
Cluster Summary:
  * Stack: corosync
  * Current DC: cl-sap-1 (version 2.0.5-9.el8_4.5-ba59be7122) - partition with quorum
  * Last updated: Tue Feb 14 09:03:19 2023
  * Last change:  Tue Feb 14 09:01:40 2023 by s01adm via crm_resource on cl-sap-1
  * 2 nodes configured
  * 11 resource instances configured

Node List:
  * Online: [ cl-sap-1 cl-sap-2 ]

Full List of Resources:
  * res_fence_ibm_powervs(stonith:fence_ibm_powervs): Started cl-sap-1
  * Resource Group: s01_ascs01_group:
    * s01_vip_ascs01(ocf::heartbeat:IPaddr2): Started cl-sap-2
    * s01_fs_ascs01_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-2
    * s01_fs_ascs01(ocf::heartbeat:Filesystem): Started cl-sap-2
    * s01_ascs01(ocf::heartbeat:SAPInstance): Started cl-sap-2
  * Resource Group: s01_ers02_group:
    * s01_vip_ers02(ocf::heartbeat:IPaddr2): Started cl-sap-1
    * s01_fs_ers02_lvm(ocf::heartbeat:LVM-activate): Started cl-sap-1
    * s01_fs_ers02(ocf::heartbeat:Filesystem): Started cl-sap-1
    * s01_ers02(ocf::heartbeat:SAPInstance): Started cl-sap-1
  * Clone Set: fs_sapmnt-clone [fs_sapmnt]:
    * Started: [ cl-sap-1 cl-sap-2 ]

Daemon Status:
  corosync: active/disabled
  Pacemaker: active/disabled
  pcsd: active/enabled
```

**Note:** Keep in mind that the ASCS resource group moved to the second node. If you run the **`pcs status`** command too soon, you might see some resources stopping and starting.

- **Test4 - Recovery Procedure**

  Wait until the ASCS instance is active on NODE2. After five minutes, the cluster removes the created location constraints automatically. Example 6-54 shows the constraints during the failover.

*Example 6-54 On NODE2 run the pcs constraint command to show constraints*

```
# pcs constraint
Location Constraints:
  Resource: s01_ascs01_group
    Constraint: cli-ban-s01_ascs01_group-on-cl-sap-1
      Rule: boolean-op=and score=-INFINITY
        Expression: #uname eq string cl-sap-1
        Expression: date lt 2023-02-08 09:33:50 -05:00
Ordering Constraints:
  start s01_ascs01_group then stop s01_ers02_group (kind:Optional) (non-symmetrical)
  start fs_sapmnt-clone then start s01_ascs01_group (kind:Mandatory)
  start fs_sapmnt-clone then start s01_ers02_group (kind:Mandatory)
Colocation Constraints:
  s01_ers02_group with s01_ascs01_group (score:-5000)
Ticket Constraints:
```

Example 6-55 shows how to remove the constraints manually by running the **`pcs resource clear`** command.

*Example 6-55 On NODE2, Clear location Constraints*

```
pcs resource clear ${sid}_ascs${ASCS_nr}_group
```

Example 6-56 shows the results of clearing the constraints.

*Example 6-56 On NODE2, run the following command.*

```
# pcs constraint
Location Constraints:
Ordering Constraints:
  start s01_ascs01_group then stop s01_ers02_group (kind:Optional) (non-symmetrical)
  start fs_sapmnt-clone then start s01_ascs01_group (kind:Mandatory)
  start fs_sapmnt-clone then start s01_ers02_group (kind:Mandatory)
Colocation Constraints:
  s01_ers02_group with s01_ascs01_group (score:-5000)
Ticket Constraints:
```

## 6.3 Using Pacemaker to create HA SAP HANA database instances

SAP HANA production databases are critical to business operations, making it essential to ensure maximum uptime through HA cluster configurations. The Red Hat HA Add-On Solutions for SAP HANA leverage Pacemaker to enable such configurations in combination with HANA System Replication (HSR).

HSR is an integrated high availability and disaster recovery feature designed to ensure business continuity by continuously replicating and synchronizing an SAP HANA database to one or more secondary locations. This setup preloads data onto the secondary system to minimize recovery time objectives (RTO).

However, while HSR keeps an up-to-date copy of the SAP HANA database, it does not automatically detect failures or manage failovers between primary and secondary systems.

The Red Hat High Availability Add-On enhances HANA HSR by providing automated failover capabilities within clusters. This add-on, built on the Red Hat Enterprise Linux High Availability cluster framework, enables automated failovers and ensures that SAP HANA databases operate within HA clusters. The Red Hat Enterprise Linux HA Add-On supplies all necessary packages to configure a Pacemaker-based cluster, offering reliability, scalability, and availability for critical SAP HANA Database services. Pacemaker's clustering functionality, when used with HANA HSR, minimizes both planned and unplanned downtime in your SAP HANA environment.

HSR offers multiple options for configuring replication between database copies, accommodating various distances between them. These options support distances ranging from local (Metro) to more extensive geographic distances, depending on network latencies and bandwidth. This flexibility enables different levels of recovery, including local recovery, metro distance recovery, and geographic-level disaster recovery for SAP HANA databases.

Figure 6-4 is an illustration of a typical HANA Database deployment with HSR capability.

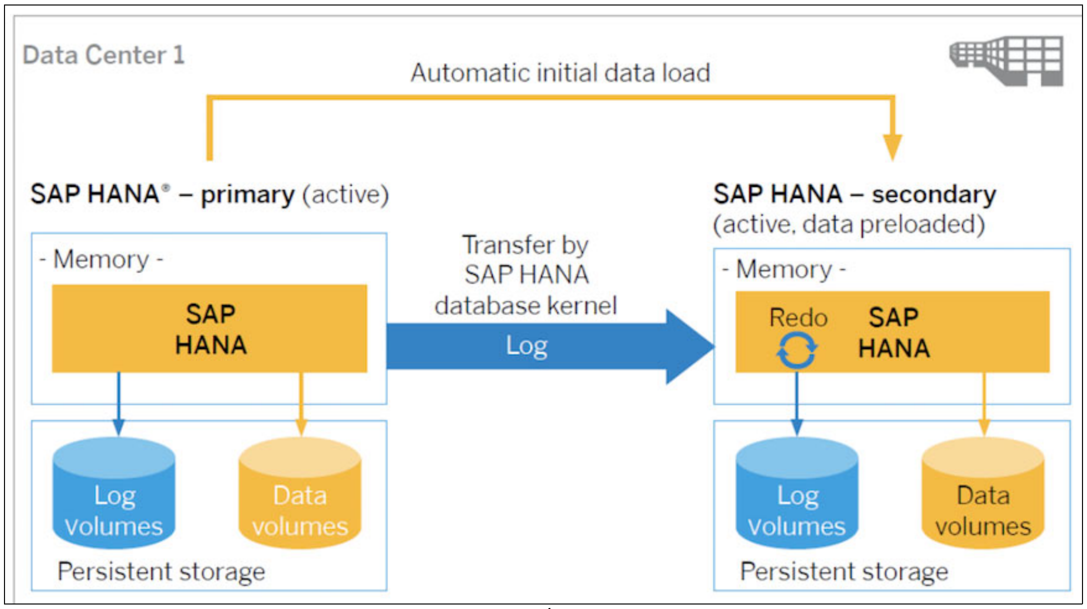


*Figure 6-4 SAP HANA System Replication (HSR)*[4]

Table 6-3 summarizes a list of all supported high availability scenarios for Automated SAP HANA Scale-Up System Replication options

*Table 6-3 Automated SAP HANA Scale-Up System Replication options*

| **Supported Scenario** | **Notes** |
|---|---|
| 6.3.1, "Performance Optimized" on page 110 | Secondary site is not active for client/application servers. |
| 6.3.2, "Cost Optimized" on page 110 | Support a QA/Test instance running on the secondary site (Cost-Optimized); QA/Test instance will be shutdown first during the failover of Prod. |

[4] Red Hat documentation on HA implementation for SAP Hana

| Supported Scenario | Notes |
|---|---|
| 6.3.3, “Active/Active (Read Enabled)” on page 111 | The secondary HANA instance can take read-only inquiries. |
| 6.3.4, “Multi-tier System Replication” on page 112 | Multi tier System Replication is possible, but the tertiary site cannot be managed by the cluster. |
| 6.3.5, “Multi-target System Replication” on page 113 | In addition to the standard HANA System Replication the data is replicated to additional secondary HANA instances that are not managed by the cluster. |

## 6.3.1 Performance Optimized

The Performance Optimized scenario, the secondary HANA database is configured to preload the tables into memory, thus the takeover time is normally very fast. However, as the secondary HANA database is dedicated to System Replication and does not accept client inquiries, this setup is expensive in terms of hardware cost.

Figure 6-5 illustrates the Performance Optimized deployment with Pacemaker cluster.

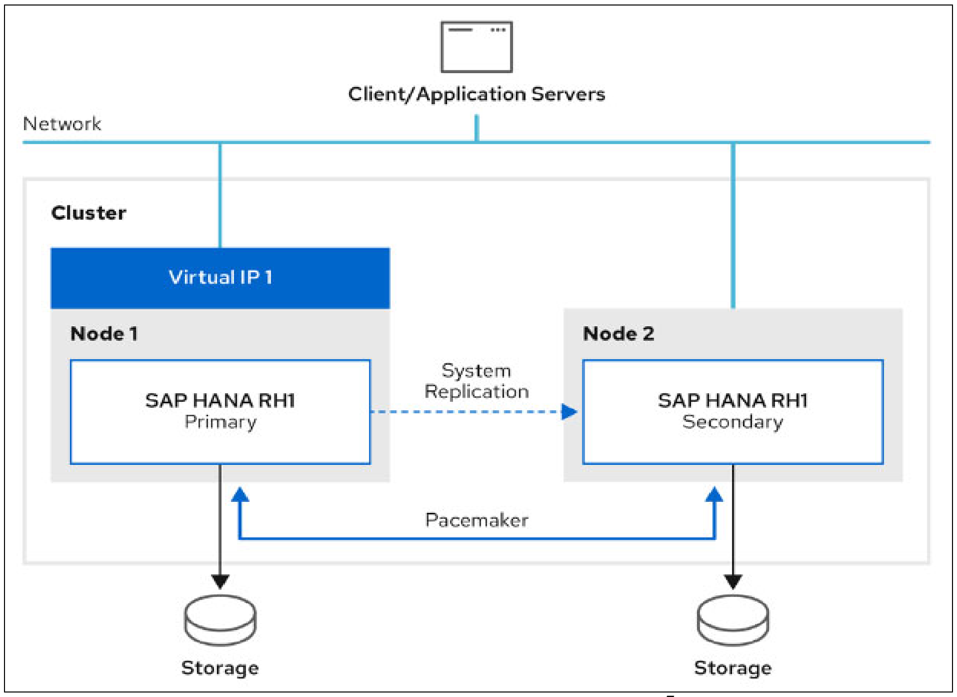


*Figure 6-5 Performance Optimized - HSR Pacemaker cluster*[5]

The configuration guide for Performance Optimized deployment can be found for IBM Power Virtual Server offering in this Configuring SAP HANA Scale-Up System Replication in a RHEL HA Add-On Cluster.

## 6.3.2 Cost Optimized

The Cost Optimized scenario provides support for additional TEST/QA HANA database on the secondary site, serving client inquiries.

[5] Red Hat HA Solutions for SAP

As the hardware resources are used and utilized by the TEST/QA instances, the Production HANA database could not be preloaded. The take over from secondary site must shutdown TEST/QA instances to allow assignment of hardware resources to the production instance so that the secondary HANA instance could be promoted as primary instance. The takeover time is effected and longer in this case for Cost Optimized deployments.

Figure 6-6 illustrates the Cost Optimized deployment with Pacemaker clusters.

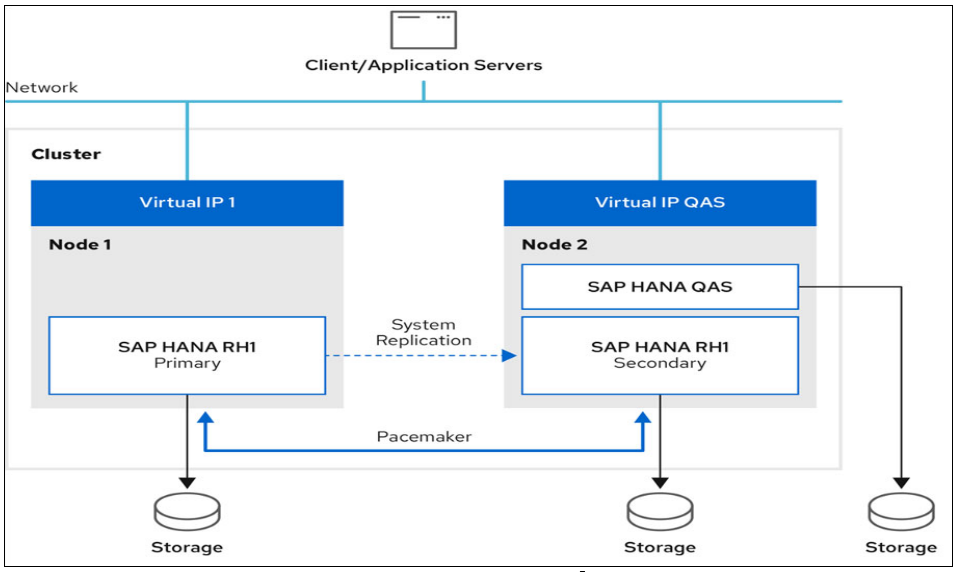


*Figure 6-6 Cost Optimized - HSR Pacemaker cluster*[6]

The configuration guide for Cost Optimized deployment with Pacemaker can be found for IBM Power Virtual Server offering in Configuring SAP HANA Cost-Optimized Scale-Up System Replication in a RHEL HA Add-On Cluster.

## 6.3.3 Active/Active (Read Enabled)

In this configuration, both primary site as well as secondary site are loaded and run in an active/active cluster where primary site instance operating in read/write mode where the secondary site instance is operating in read-enabled mode where secondary instance is able to service read only queries over a second Virtual IP address. Figure 6-7 on page 112 shows the Active/Active (Read Enabled) deployment with Pacemaker clusters.

[6] Red Hat HA Solutions for SAP

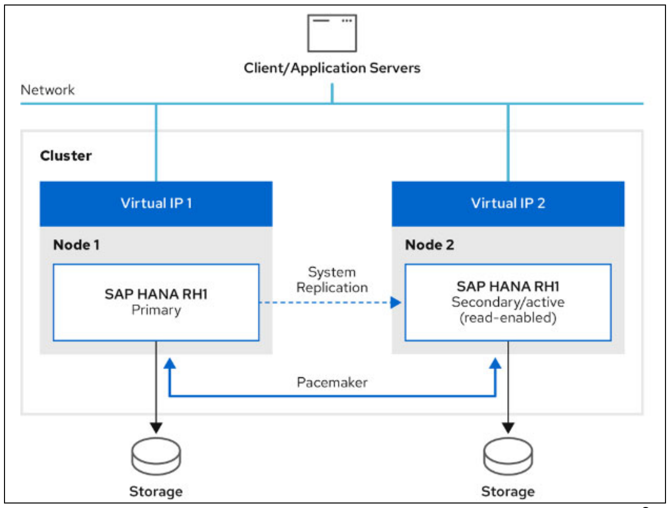


*Figure 6-7 Active/Active (read-enabled) Pacemaker cluster*[6]

The configuration guide for Active/Active (read-enabled) deployment with Pacemaker can be found for IBM Power Virtual Server offering in Configuring SAP HANA Active/Active (Read Enabled) System Replication in  a RHEL HA Add-On Cluster.

### 6.3.4 Multi-tier System Replication

Multi-tier System Replication provides a HANA database to be replicated using HSR from primary site to Secondary and then onto a third site as a chain replication. In this option, any failovers or recovery activities are totally manual for the third site. The Primary and Secondary sites with HANA databases are setup to automatically fail-over. However, the Pacemaker cluster will not have any configuration or the knowledge of the third site. The fail-back of the cluster need to be manually performed with the cluster stopped as there is no knowledge of the third site in the Pacemaker cluster.

Figure 6-8 shows the Multi-tier System Replication deployment with Pacemaker clusters.

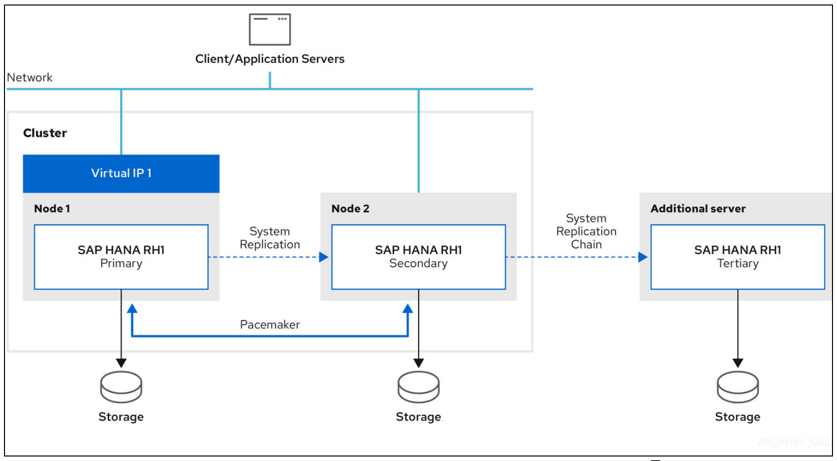


*Figure 6-8 Multi-tier System Replication Pacemaker Cluster*[7]

[7] Red Hat HA Solutions for SAP

The Configuration guide for Multi-tier System Replication deployment with Pacemaker can be found for IBM Power Virtual Server offering in Configuring SAP HANA Multitier System Replication in a RHEL HA Add-On Cluster.

### 6.3.5 Multi-target System Replication

The Multi-target System Replication HA cluster setup in Scale up deployment of HANA 2.0 databases, the primary HANA instance is replicated to a secondary HANA instance managed by the HA cluster and also, primary instance is asynchronously to an additional secondary HANA instances that is not managed by the cluster to meet additional HA-DR capabilities.

Figure 6-9 shows the Multi-target System Replication deployment with Pacemaker clusters.

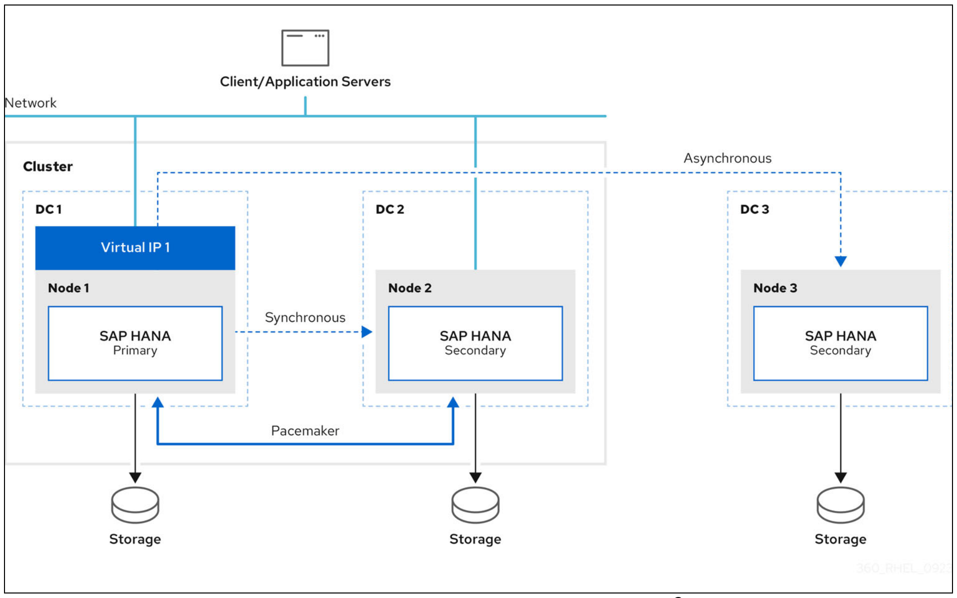


*Figure 6-9 Multi-target System Replication Pacemaker cluster*[8]

Configuration guide for Multi-target System Replication deployment with Pacemaker can be found for on-premise deployments in this Red Hat Document.

**Note:** This section is not meant to replace any SAP or Red Hat Documentation. You should always refer to respective SAP or Red Hat Documentation for latest and current information.

### 6.3.6 Configuring SAP HANA Scale-Up System Replication in a RHEL HA Add-On cluster

#### Overview

This section describes the configuration of a Red Hat Enterprise Linux 8 (RHEL) HA Add-On cluster for managing SAP HANA Scale-Up System Replication. The cluster uses virtual server instances within IBM Power Virtual Server offering in IBM Cloud as cluster nodes.

[8] Red Hat HA Solutions for SAP

## Prerequisites

1. A Red Hat High Availability cluster deployed on two virtual server instances in Power Systems Virtual Server.
   a. Install and set up the RHEL HA Add-On cluster
   b. Configure and verify fencing as described in the preceding sections
2. The virtual server instances used from IBM Power Virtual Server offering need to fulfill hardware and resource requirements for the SAP HANA systems within its scope.
3. The hostnames used for the virtual server instances must meet the SAP HANA requirement.
4. SAP HANA installed on both virtual server instances and SAP HANA System Replication is configured following standard SAP HANA installation procedures for IBM Power Systems. The following references describe this in detail:
   a. Install SAP HANA on RHEL:
      - SAP Note 2772999 - Red Hat Enterprise Linux 8.x: Installation and Configuration
      - SAP Note 2777782 - SAP HANA DB: Recommended OS Settings for RHEL 8
   b. SAP HANA Server Installation:
      - SAP HANA Server Installation and Update Guide
   c. SAP HANA System Replication:
      - SAP HANA System Replication Guide
   d. Configure SAP HANA System Replication:
      - Automating SAP HANA Scale-Up System Replication by using the RHEL HA Add-On
      - SAP HANA System Replication
5. A valid RHEL for SAP Applications or RHEL for SAP Solutions subscription is required to enable the repositories that you need to install SAP HANA and the resource agents for HA configurations.

For more information, see Automating SAP HANA Scale-Up System Replication by using the RHEL HA Add-On.

## Environment variables

To simplify the setup and configuration, on both nodes, as root, set the environment variables shown in Example 6-57 as they are used throughout the following example.

*Example 6-57 Environment Variables*

```
export SID=<SID>             # SAP HANA System ID (uppercase)
export sid=<sid>             # SAP HANA System ID (lowercase)
export INSTNO=<INSTNO>       # SAP HANA Instance Number

export DC1=<Site1>           # HANA System Replication Site Name 1
export DC2=<Site2>           # HANA System Replication Site Name 2

export NODE1=<Hostname 1>    # Hostname of virtual server instance 1
export NODE2=<Hostname 2>    # Hostname of virtual server instance 2
```

## Installing SAP HANA Resource Agents

Run the command shown in Example 6-58 on page 115 to install the RHEL HA Add-On resource agents for SAP HANA System Replication.

*Example 6-58 Install SAP HANA Resource Agents*

```
dnf install -y resource-agents-sap-hana
```

## Starting SAP HANA System

Start SAP HANA and verify that HANA System Replication is active. On both nodes, run the commands shown in Example 6-59. For more information, refer to "Checking SAP HANA System Replication state".

*Example 6-59 Start HANA and Check Status*

```
sudo -i -u ${sid}adm -- HDB start

sudo -i -u ${sid}adm -- <<EOT
    hdbnsutil -sr_state
    HDBSettings.sh systemReplicationStatus.py
EOT
```

## Enabling the SAP HANA srConnectionChanged() hook

Recent versions of SAP HANA provide hooks so SAP HANA can send out notifications for certain events. For more information, refer to "Implementing a HA/DR Provider".

The **`srConnectionChanged()`** hook improves the ability of the cluster to detect a status change of HANA System Replication that requires an action from the cluster. The goal is to prevent data loss and corruption by preventing accidental takeovers.

### Activating the srConnectionChanged() hook on all SAP HANA instances

First, stop the cluster as shown in Example 6-60.

*Example 6-60 Stop the Cluster*

```
# On NODE1, run the following command.
pcs cluster stop --allEOT
```

Now, install the hook script. Run the commands shown in Example 6-61 on each HANA node in the cluster.

*Example 6-61 Install the hook script for each HANA instance, and set the required ownership.*

```
# On Both Nodes, run the following commands.

mkdir -p /hana/shared/myHooks
cp /usr/share/SAPHanaSR/srHook/SAPHanaSR.py /hana/shared/myHooks
chown -R ${sid}adm:sapsys /hana/shared/myHooks
```

Update the `global.ini` file one each HANA node as seen in Example 6-62. This enables the hook script.

*Example 6-62 Update the **global.ini** file on each HANA node to enable the hook script.*

```
# On Both Nodes, run the following command.

sudo -i -u ${sid}adm -- <<EOT
  python \$DIR_INSTANCE/exe/python_support/setParameter.py \
    -set SYSTEM/global.ini/ha_dr_provider_SAPHanaSR/provider=SAPHanaSR \
    -set SYSTEM/global.ini/ha_dr_provider_SAPHanaSR/path=/hana/shared/myHooks \
    -set SYSTEM/global.ini/ha_dr_provider_SAPHanaSR/execution_order=1 \
    -set SYSTEM/global.ini/trace/ha_dr_saphanasr=info
EOT
```

Validate the changes to the `global.ini` file using the command in Example 6-63.

*Example 6-63 Verify the changed* ***global.ini*** *file.*

```
# On Both Nodes, run the following commands.
cat /hana/shared/${SID}/global/hdb/custom/config/global.ini
```

Create sudo settings for SAP HANA OS user which is required to allow the right permissions for the user during configuration changes. See Example 6-64.

*Example 6-64 Create sudo settings for SAP HANA OS user*

```
# You need the following sudo settings to allow the ${sid}adm user
# Script can update the node attributes when the srConnectionChanged()
# hook runs.
#
# Create a file with the required sudo aliases and user specifications.
# On Both Nodes, run the following commands.

cat >> /etc/sudoers.d/20-saphana << EOT
Cmnd_Alias DC1_SOK = /usr/sbin/crm_attribute -n hana_${sid}_site_srHook_${DC1} -v SOK -t
crm_config -s SAPHanaSR
Cmnd_Alias DC1_SFAIL = /usr/sbin/crm_attribute -n hana_${sid}_site_srHook_${DC1} -v SFAIL
-t crm_config -s SAPHanaSR
Cmnd_Alias DC2_SOK = /usr/sbin/crm_attribute -n hana_${sid}_site_srHook_${DC2} -v SOK -t
crm_config -s SAPHanaSR
Cmnd_Alias DC2_SFAIL = /usr/sbin/crm_attribute -n hana_${sid}_site_srHook_${DC2} -v SFAIL
-t crm_config -s SAPHanaSR
${sid}adm ALL=(ALL) NOPASSWD: DC1_SOK, DC1_SFAIL, DC2_SOK, DC2_SFAIL
Defaults!DC1_SOK, DC1_SFAIL, DC2_SOK, DC2_SFAIL !requiretty
EOT

# Adjust the permissions and check for syntax errors.

chown root:root /etc/sudoers.d/20-saphana
chmod 0440 /etc/sudoers.d/20-saphana
cat /etc/sudoers.d/20-saphana
visudo -c
```

**Important:** Any problems that are reported by the **`visudo -c`** command must be corrected.

Verify that the configured hook works:

– Restart both HANA instances and verify that the hook script works as expected.
– Perform an action to trigger the hook, such as stopping a HANA instance.
– Check whether the hook logged anything in the trace files.

These actions are shown in Example 6-65.

*Example 6-65 Restart and then stop the HANA instance.*

```
# On Both Nodes, run the following commands.

sudo -i -u ${sid}adm -- HDB restart
sudo -i -u ${sid}adm -- HDB stop
```

```
# Check messages in trace files.

sudo -i -u ${sid}adm -- sh -c 'grep "ha_dr_SAPHanaSR.*crm_attribute"
$DIR_INSTANCE/$VTHOSTNAME/trace/nameserver_* | cut -d" " -f2,3,5,17'

# Start the HANA instance.

sudo -i -u ${sid}adm -- HDB start
```

**Note:** After you verify that the hooks function, you can restart the HA cluster.

Start the cluster using the commands shown in Example 6-66.

*Example 6-66 Start cluster.*

```
# On NODE1, run the following commands.
pcs cluster start --all

# Check the status of the cluster.
pcs status --full
```

## Configuring general cluster properties

To avoid failovers of the resources during initial testing and post production, set the following default values (shown in Example 6-67) for the resource-stickiness and migration-threshold parameters. Keep in mind that defaults do not apply to resources that override them with their own defined values.

*Example 6-67* Start cluster.

```
# On NODE1, run the following commands.
pcs resource defaults update resource-stickiness=1000
pcs resource defaults update migration-threshold=5000
```

## Creating a cloned SAPHanaTopology resource

The SAPHanaTopology resource gathers status and configuration of SAP HANA System Replication on each node. It also starts and monitors the local SAP Host Agent, which is required for starting, stopping, and monitoring SAP HANA instances. Create this resource with the commands shown in Example 6-68.

*Example 6-68 Create the SAPHanaTopology resource.*

```
# On NODE1, run the following commands.

pcs resource create SAPHanaTopology_${SID}_${INSTNO} SAPHanaTopology \
  SID=${SID} InstanceNumber=${INSTNO} \
  op start timeout=600 \
  op stop timeout=300 \
  op monitor interval=10 timeout=600 \
  clone clone-max=2 clone-node-max=1 interleave=true

# Check the configuration and the cluster status by running the
# following commands.

pcs resource config SAPHanaTopology_${SID}_${INSTNO}
pcs resource config SAPHanaTopology_${SID}_${INSTNO}-clone
pcs status --full
```

## Creating master and slave SAPHana resources

The SAPHana resource manages two SAP HANA instances that are configured as HANA System Replication nodes. Create these resources as shown in Example 6-69.

*Example 6-69 Create the SAPHana resource*

```
# On NODE1, run the following commands.
pcs resource create SAPHana_${SID}_${INSTNO} SAPHana \
  SID=${SID} InstanceNumber=${INSTNO} \
  PREFER_SITE_TAKEOVER=true \
  DUPLICATE_PRIMARY_TIMEOUT=7200 \
  AUTOMATED_REGISTER=false \
  op start timeout=3600 \
  op stop timeout=3600 \
  op monitor interval=61 role="Slave" timeout=700 \
  op monitor interval=59 role="Master" timeout=700 \
  op promote timeout=3600 \
  op demote timeout=3600 \
  promotable notify=true clone-max=2 clone-node-max=1 interleave=true

# Check the configuration and the cluster status.
pcs resource config SAPHana_${SID}_${INSTNO}
pcs status --full
```

## Creating a virtual IP address resource

Review the information in Reserving virtual IP addresses and reserve a virtual IP address for the SAP HANA System Replication cluster.

Use the reserved IP address to create a virtual IP address resource. This virtual IP address is used to reach the System Replication primary instance.

Assign the reserved IP address to a VIP environment variable and create the virtual IP address cluster resource as shown in Example 6-70.

*Example 6-70 Create VIP*

```
# On NODE1, run the following commands.
export VIP=<reserved IP address>
echo $VIP
pcs resource create vip_${SID}_${INSTNO} IPaddr2 ip=$VIP

# Check the configured virtual IP address and the cluster status.
pcs resource config vip_${SID}_${INSTNO}
ip addr show

pcs status --full
```

## Creating constraints

Make sure that “SAPHanaTopology” resources are started before you start the “SAPHana” resources.

The virtual IP address must be present on the node where the primary resource of "SAPHana" is running. This constraint mandates the start order of these resources. Example 6-71 on page 119 shows the order constraint being set.

*Example 6-71 Create the SAPHanaTopology order constraint.*

```
# On NODE1, run the following commands.
pcs constraint order SAPHanaTopology_${SID}_${INSTNO}-clone \
  then SAPHana_${SID}_${INSTNO}-clone symmetrical=false

# Check the configuration.
pcs constraintip addr show
```

Example 6-72 shows setting the collocation constraint.

*Example 6-72 Create constraint to collocate the virtual IP address with primary.*

```
# On NODE1, run the following commands.
pcs constraint colocation add vip_${SID}_${INSTNO} \
  with master SAPHana_${SID}_${INSTNO}-clone 2000

# Check the configuration and the cluster status.
pcs constraint
pcs status --full
```

**Note:** This constraint collocates the virtual IP address resource with the SAPHana resource that was promoted as primary.

## Enabling automated registration of secondary instance

You need to set the parameter AUTOMATED_REGISTER according to your operational requirements. If you want to keep the ability to revert to the state of the previous primary SAP HANA instance, then AUTOMATED_REGISTER=false avoids an automatic registration of the previous primary as a new secondary.

If you experience an issue with the data after a takeover that was triggered by the cluster, you can manually revert if AUTOMATED_REGISTER is set to false.

If AUTOMATED_REGISTER is set to true, the previous primary SAP HANA instance automatically registers as secondary, and cannot be activated on its previous history. The advantage of AUTOMATED_REGISTER=true is that HA capability is automatically reestablished after the failed node reappears in the cluster.

For now, it is recommended to keep AUTOMATED_REGISTER set to the default value – false – until the cluster is fully tested and that you verify that the failover scenarios work as expected.

**Tip:** The **`pcs resource update`** command is used to modify resource attributes and **`pcs resource update SAPHana_${SID}_${INSTNO} AUTOMATED_REGISTER=true`** sets the attribute to true.

## Testing the SAP HANA System Replication cluster

It is vital to thoroughly test the cluster configuration to make sure that the cluster is working correctly. The following information provides a few sample failover test scenarios, but is not a complete list of test scenarios.

For example, the description of each test case includes the following information:

- Component that is being tested.
- Description of the test.
- Prerequisites and the cluster state before you start the failover test.

- Test procedure.
- Expected behavior and results.
- Recovery procedure.

### Test1 - Testing failure of the primary database instance

Use the proceeding information to test the failure of the primary database instance.

- **Test1 - Description**

This test will simulate a failure of the primary HANA database instance that is running on NODE1.

- **Test1 - Prerequisites**
  - A functional two-node RHEL HA Add-On cluster for HANA system replication.
  - Both cluster nodes are active.
  - Cluster that is started on NODE1 and NODE2.
  - Cluster Resource SAPHana_${SID}_${INSTNO} that is configured with AUTOMATED_REGISTER=false.
  - Check SAP HANA System Replication status:
  - Primary SAP HANA database is running on NODE1
  - Secondary SAP HANA database is running on NODE2
  - HANA System Replication is activated and in sync
- **Test1 - Test procedure**

Forcefully terminate the SAP HANA primary instance by sending a SIGKILL signal as user ${sid}adm. This is shown in Example 6-73.

*Example 6-73 Kill the primary HANA database*

```
# On NODE1, run the following commands.
sudo -i -u ${sid}adm -- HDB kill-9
```

- **Test1 - Expected behavior**
  - SAP HANA primary instance on NODE1 fails.
  - The cluster detects the stopped primary HANA database and marks the resource as failed.
  - The cluster promotes the secondary HANA database on NODE2 to take over as new primary.
  - The cluster releases the virtual IP address on NODE1, and acquires it on the new primary on NODE2.
  - If an application, such as SAP NetWeaver, is connected to a tenant database of SAP HANA, the application automatically reconnects to the new primary.
- **Test1 - Recovery procedure**

As cluster resource SAPHana_${SID}_${INSTNO} is configured with the setting AUTOMATED_REGISTER=false, the cluster doesn't restart the failed HANA database, and does not register it against the new primary. Which means that the status on the new primary (NODE2) also shows the secondary in status CONNECTION TIMEOUT. To recover register the previous primary database as a secondary as shown in Example 6-74.

*Example 6-74 Re-register the previous primary as new secondary*

```
# On NODE1, run the following commands.

sudo -i -u ${sid}adm -- <<EOT
  hdbnsutil -sr_register \
  --name=${DC1} \
  --remoteHost=${NODE2} \
  --remoteInstance=00 \
  --replicationMode=sync \
  --operationMode=logreplay
EOT
#
# Verify the system replication status:
#
sudo -i -u ${sid}adm -- <<EOT
  hdbnsutil -sr_state
  HDBSettings.sh systemReplicationStatus.py
EOT
#
# On NODE1, run the following commands.
#
pcs resource refresh SAPHana_${SID}_${INSTNO}
pcs status --full
```

**Note:** After the manual register and resource refreshes, the new secondary instance restarts and shows up in status synced (SOK).

## Test2 - Testing failure of the node that is running the primary database

Use the following information to test the failure of the node that is running the primary database.

- **Test2 - Description**

This test will simulate a failure of the node that is running the primary HANA database.

- **Test2 - Preparation**

Make sure that Cluster Resource SAPHana_${SID}_${INSTNO} is configured with AUTOMATED_REGISTER=true. This is shown in Example 6-75.

*Example 6-75 On NODE1, run the following command.*

```
pcs resource update SAPHana_${SID}_${INSTNO} AUTOMATED_REGISTER=true
pcs resource config SAPHana_${SID}_${INSTNO}
```

- **Test2 - Prerequisites**
  - A functional two-node RHEL HA Add-On cluster for HANA system replication.
  - Both nodes active.
  - Cluster is started on NODE1 and NODE2.
  - Check SAP HANA System Replication status.
  - Primary SAP HANA database is running on NODE2.
  - Secondary SAP HANA database is running on NODE1.
  - HANA System Replication is activated and in sync.
- **Test2 - Test procedure**

Forcefully terminate the primary on NODE2 by sending a shutoff system request as shown in Example 6-76.

*Example 6-76 On NODE2, run the following command.*

```
sync; echo o > /proc/sysrq-trigger
```

- **Test2 - Expected behavior**
  - NODE2 shuts down.
  - The cluster detects the failed node and sets its state to OFFLINE.
  - The cluster promotes the secondary HANA database on NODE1 to take over as new primary.
  - The cluster acquires the virtual IP address on NODE1 on the new primary.
  - If an application, such as SAP NetWeaver, is connected to a tenant database of SAP HANA, the application automatically reconnects to the new primary.
- **Test2 - Recovery procedure**

Log in to the IBM Cloud Console and start the NODE2 instance. Wait until NODE2 is available again, then restart the cluster framework. This is shown in Example 6-77.

*Example 6-77 On NODE2, run the following commands.*

```
pcs cluster start
pcs status --full
```

**Note:** As cluster resource SAPHana_${SID}_${INSTNO} is configured with AUTOMATED_REGISTER=true, SAP HANA restarts when NODE2 rejoins the cluster and the former primary re-registers as a secondary.

## Test3 - Testing the failure of the secondary database instance

Use the following information to test the failure of the secondary database instance.

- **Test3 - Description**

Simulate a failure of the secondary HANA database.

- **Test3 - Prerequisites**
  - A functional two-node RHEL HA Add-On cluster for HANA system replication.
  - Both nodes are active.
  - Cluster is started on NODE1 and NODE2.
  - Cluster Resource SAPHana_${SID}_${INSTNO} is configured with AUTOMATED_REGISTER=true.
  - Check SAP HANA System Replication status:
    - Primary SAP HANA database is running on NODE1
    - Secondary SAP HANA database is running on NODE2
    - HANA System Replication is activated and in sync
- **Test3 - Test Procedure**

Forcefully terminate the SAP HANA secondary instance by sending a SIGKILL signal as user ${sid}adm. On NODE2, run the command shown in Example 6-78.

*Example 6-78 Kill secondary HANA database*

```
sudo -i -u ${sid}adm -- HDB kill-9
```

- **Test3 - Expected behavior**
  - SAP HANA secondary on NODE2 stops.

  - The cluster detects the stopped secondary HANA database and marks the resource as failed.
  - The cluster restarts the secondary HANA database.
  - The cluster detects that the system replication is in sync again.
- **Test3 - Recovery procedure**

Wait until the secondary HANA instance starts and syncs again (SOK), then cleanup the failed resource actions as shown in **`pcs status`**. This is shown in Example 6-79.

*Example 6-79 On NODE2, run the following command.*

```
pcs resource refresh SAPHana_${SID}_${INSTNO}
pcs status --full
```

## Test4 - Testing the manual move of a SAPHana resource to another node

This test will manually move a SAPHana resource to another node.

- **Test4 - Description**

Use cluster commands to move the primary instance to the other node for maintenance purposes.

- **Test4 - Prerequisites**
  - A functional two-node RHEL HA Add-On cluster for HANA system replication.
  - Both nodes are active.
  - Cluster is started on NODE1 and NODE2.
  - Cluster Resource SAPHana_${SID}_${INSTNO} is configured with AUTOMATED_REGISTER=true.
  - Check SAP HANA System Replication status:
    - Primary SAP HANA database is running on NODE1.
    - Secondary SAP HANA database is running on NODE2.
    - HANA System Replication is activated and in sync.

### *Test4 - Test procedure*

Move SAP HANA primary to other node by using the **`pcs resource move`** command as shown in Example 6-80.

*Example 6-80 Move SAP primary*

```
pcs resource move SAPHana_${SID}_${INSTNO}-clone
```

- **Test4 - Expected behavior**
  - The cluster creates location constraints to move the resource.
  - The cluster triggers a takeover to the secondary HANA database.
  - If an application, such as SAP NetWeaver, is connected to a tenant database of SAP HANA, the application automatically reconnects to the new primary.
- **Test4 - Recovery procedure**

The automatically created location constraints must be removed to allow automatic failover in the future. Wait until the primary HANA instance is active and remove the constraints. when the restraints are removed, the cluster registers and starts the HANA database as a new secondary instance. This is shown in Example 6-81 on page 124.

*Example 6-81 Remove constraints*

```
pcs constraint
pcs resource clear SAPHana_${SID}_${INSTNO}-clone
pcs constraint
pcs status --full
```

7

# Additional Use Cases for Pacemaker HA

Pacemaker stands as a cornerstone in various scenarios ensuring system resilience and continuity. This chapter delves into additional use cases where Pacemaker plays a pivotal role in maintaining high availability (HA) within diverse environments. While these use cases have undergone testing and scrutiny within this publication, caution and meticulous attention are advised when implementing these solutions in production environments. Each scenario detailed here showcases the potential of Pacemaker but warrants careful consideration and validation to ensure seamless deployment and operation in live, production-grade settings.

This chapter describes the following topics:

- 7.1, "IBM MQ Replicated Data Queue Manager" on page 126
- 7.2, "Using VM Recovery Manager to Replicate Pacemaker cluster" on page 132
- 7.3, "NSF Server HA" on page 138
- 7.4, "IBM MQ active/passive architecture using Pacemaker and NFS on IBM Power Virtual Server" on page 139
- 7.5, "IBM PowerVC three-node architecture using Pacemaker" on page 140

## 7.1 IBM MQ Replicated Data Queue Manager

A Replicated Data Queue Manager (RDQM) setup consists of a trio of servers meticulously configured into an HA group, each hosting an instance of the queue manager. Among these instances, one serves as the active queue manager, diligently replicating its data to the other two instances in real-time. In the unfortunate event of the server hosting the active queue manager encountering an issue, another instance of the queue manager swiftly takes over, ensuring seamless operation with up-to-date data.

Furthermore, the three queue manager instances can opt to share a floating IP address, simplifying client configurations by requiring just a single IP address. Importantly, even if the HA group experiences network disruptions leading to a partition, only one instance of the queue manager can run concurrently. The server responsible for running the queue manager is referred to as the primary, while the other two servers are known as secondaries.

To avert the possibility of a split-brain scenario, this configuration employs three nodes. In a two-node HA system, the risk of split-brain arises when the connection between the two nodes is severed. In such a situation, both nodes could potentially run the queue manager simultaneously, resulting in the accumulation of disparate data sets. When connectivity is eventually restored, these conflicting data versions create a split-brain, necessitating manual intervention to determine which dataset to retain and which to discard.

**IMPORTANT:** The RDQM (Replicated Data Queue Manager) component is supported solely on Linux for x86-64 (64 bit) on RHEL 7.3 or later versions.

However, for the purpose of demonstrating Pacemaker functionality, we are configuring MQ RDQM on Red Hat Linux running on IBM Power. It is crucial to emphasize that this setup is intended solely for demonstration purposes and does not imply any support for the configuration.

For detailed and updated information, we strongly recommend referring to the latest IBM MQ documentation.

### Distributed Replicated Block Device

A Distributed Replicated Block Device, often abbreviated as DRBD, is a sophisticated storage technology designed to ensure data redundancy, HA, and fault tolerance within a distributed computing environment.

At its core, DRBD operates by mirroring data blocks across multiple storage devices or nodes in a network. Each node typically hosts a DRBD resource, and these resources are kept in sync through real-time replication. This replication can occur synchronously or asynchronously, depending on the specific configuration and requirements of the system:

- DRBD ensures that data is duplicated across multiple nodes, providing a safety net in case of hardware failures or data corruption. This redundancy minimizes the risk of data loss.
- DRBD contributes to HA solutions by enabling failover capabilities. In the event of a node failure, another node can seamlessly take over, ensuring uninterrupted access to the stored data.
- DRBD is designed to provide good performance by optimizing data transfer and replication mechanisms. It strives to minimize latency and maximize throughput, making it suitable for various demanding applications.

- DRBD can be configured in various ways to suit specific use cases. Administrators can choose between synchronous and asynchronous replication modes, adjust replication policies, and configure storage layouts to meet their performance and availability requirements.
- It is commonly used with Linux-based systems and integrates well with existing Linux storage management tools, making it a popular choice for organizations that rely on Linux servers.
- DRBD can be used in clusters with multiple nodes, allowing organizations to scale their storage infrastructure to accommodate growing data needs.
- DRBD includes features to ensure data consistency and integrity, such as checksums and error handling mechanisms.

**Note:** DRBD 9 undergoes testing for compatibility on various CPU architectures, including:

- amd64
- arm64
- ppc64le
- s390x

Recent iterations of DRBD 9 are exclusively verified for construction on 64-bit CPU architectures.

Note that DRBD functionality is integrated into the MQ RDQM architecture. Figure 7-1 depicts a standard configuration where there is an RDQM operating on each of the three nodes within the HA group. In the normal mode, one of the nodes acts as primary and the other two nodes act as secondary nodes.

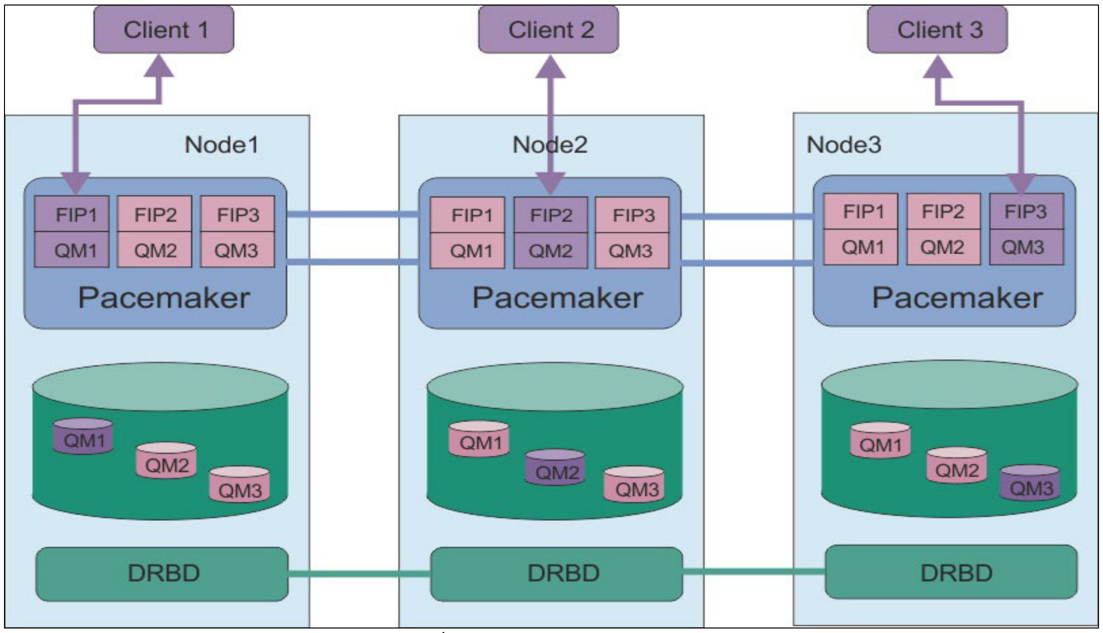


*Figure 7-1 Normal RDQM configuration*[1]

[1] RDQM high availability

In Figure 7-2 we show where Node 3 experiences a failure, resulting in the loss of Pacemaker links, and consequently, queue manager QM3 is now running on Node 2.

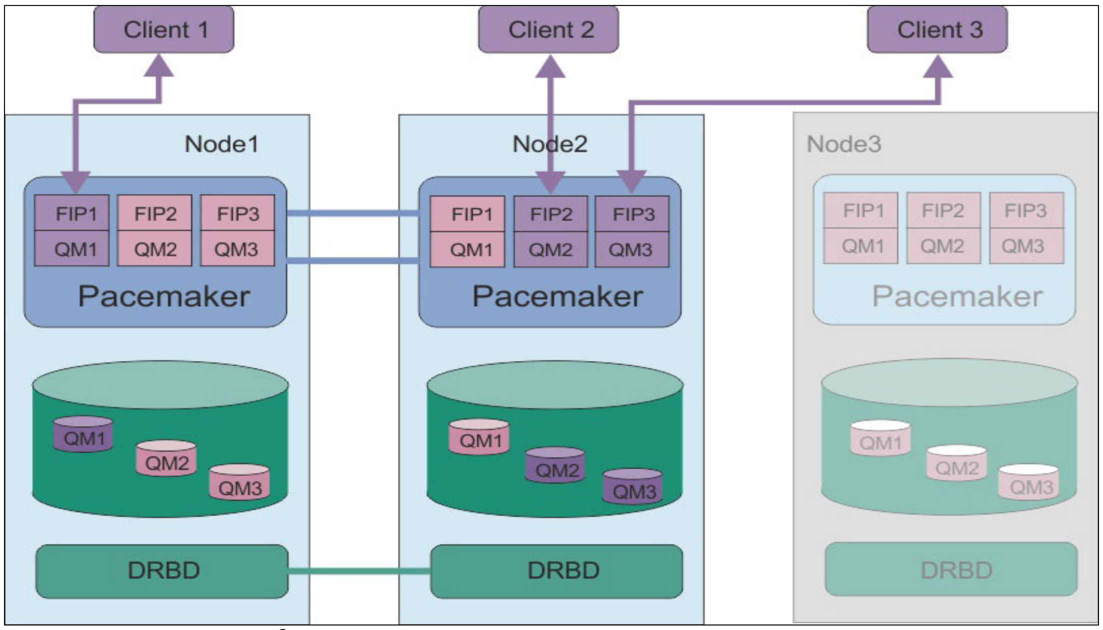


*Figure 7-2 Node 3 failed*[2]

## Configuring IBM MQ RDQM

To configure IBM MQ RDQM follow these instructions:

- Verify the Red Hat Enterprise Linux version using the command shown in Example 7-1.

*Example 7-1 Verifying OS version*

```
$ hostnamectl
 Static hostname: p1325-pvm1
       Icon name: computer-vm
         Chassis: vm ?
      Machine ID: 500ede37aabe4bc4bf54cd41745ec3fa
         Boot ID: 50dd26cdedc5471e923472a91ca8666a
  Virtualization: powervm
Operating System: Red Hat Enterprise Linux 9.2 (Plow)
     CPE OS Name: cpe:/o:redhat:enterprise_linux:9::baseos
          Kernel: Linux 5.14.0-284.11.1.el9_2.ppc64le
    Architecture: ppc64-le
```

- List available repositories using the command shown in Example 7-2.

*Example 7-2 List available repositories*

```
$ subscription-manager repos --list
You are attempting to run "subscription-manager" which requires administrative
privileges, but more information is needed in order to do so.
Authenticating as "root"
Password:
+----------------------------------------------------------+
    Available Repositories in /etc/yum.repos.d/redhat.repo
+----------------------------------------------------------+
```

[2] RDQM high availability

```
Repo ID:   rhel-9-for-ppc64le-highavailability-rpms
Repo Name: Red Hat Enterprise Linux 9 for Power, little endian - HA (RPMs)
Repo URL:
https://rhns.pbm.ihost.com/pulp/content/Default_Organization/Library/content/dist/rhel9/$re
le
           asever/ppc64le/highavailability/os
Enabled:   0

Repo ID:   codeready-builder-for-rhel-9-ppc64le-rpms
Repo Name: Red Hat CodeReady Linux Builder for RHEL 9 Power, little endian (RPMs)
Repo URL:
https://rhns.pbm.ihost.com/pulp/content/Default_Organization/Library/content/dist/rhel9/$re
le
           asever/ppc64le/codeready-builder/os
Enabled:   1

Repo ID:   rhel-9-for-ppc64le-baseos-rpms
Repo Name: Red Hat Enterprise Linux 9 for Power, little endian - BaseOS (RPMs)
Repo URL:
https://rhns.pbm.ihost.com/pulp/content/Default_Organization/Library/content/dist/rhel9/$re
le
           asever/ppc64le/baseos/os
Enabled:   1

Repo ID:   rhel-9-for-ppc64le-sap-solutions-rpms
Repo Name: Red Hat Enterprise Linux 9 for Power, little endian - SAP Solutions (RPMs)
Repo URL:
https://rhns.pbm.ihost.com/pulp/content/Default_Organization/Library/content/dist/rhel9/$re
le
           asever/ppc64le/sap-solutions/os
Enabled:   0

Repo ID:   rhel-9-for-ppc64le-appstream-rpms
Repo Name: Red Hat Enterprise Linux 9 for Power, little endian - AppStream (RPMs)
Repo URL:
https://rhns.pbm.ihost.com/pulp/content/Default_Organization/Library/content/dist/rhel9/$re
le
           asever/ppc64le/appstream/os
Enabled:   1

Repo ID:   rhel-9-for-ppc64le-supplementary-rpms
Repo Name: Red Hat Enterprise Linux 9 for Power, little endian - Supplementary (RPMs)
Repo URL:
https://rhns.pbm.ihost.com/pulp/content/Default_Organization/Library/content/dist/rhel9/$re
le
           asever/ppc64le/supplementary/os
Enabled:   1
```

- Enable the Red Hat HA repository using the command shown in Example 7-3.

*Example 7-3 Enable Red Hat HA Repository*

```
$ sudo subscription-manager repos --enable=rhel-9-for-ppc64le-highavailability-rpms
Repository 'rhel-9-for-ppc64le-highavailability-rpms' is enabled for this system.
```

- Download and install DRDB9 Binary Packages as described in LINBIT Supplied Packages.

Before proceeding with the setup of the RDQM HA group, specific configurations are necessary on each of the three servers designated for inclusion in the HA group. On each node, the following prerequisites must be met:

- A volume group named `drbdpool` should be created.
- Each node should be equipped with up to three interfaces, each serving distinct purposes within the RDQM setup:
  - A primary interface dedicated to Pacemaker for monitoring the HA group.
  - An alternate interface intended for Pacemaker to monitor the HA group.
  - A designated replication interface used for synchronous data replication.
- Additionally, if a firewall exists between the nodes participating in the HA group, it is imperative to ensure that the firewall allows traffic to flow seamlessly between the nodes across a specified range of ports.
- In cases where the system employs SUSE Enterprise Linux in Enforcing mode, it is essential to execute the command shown in Example 7-4 for proper configuration.

*Example 7-4 semanage command*

```
$ semanage permissive -a drbd_t
```

**Note:** Should you decide to deploy the nodes across different data centers, it is crucial to take into consideration the following constraints:

- Performance experiences a rapid deterioration as latency between data centers increases. While IBM provides support for latency of up to 5 ms, you may discover that your application's performance becomes intolerant when latency exceeds 1 to 2 ms.
- Data transmitted across the replication link does not undergo any additional encryption beyond what may already be in place through the use of IBM MQ AMS.

In the subsequent sections, we configure the RDQM HA group with the root user. Employ a user within the mqm group, equipped with the appropriate permissions. Detailed instructions for setting up the user mqm with the correct permissions can be found at Requirements for RDQM HA solution.

- You have the option to establish SSH access without the need for passwords, allowing you to issue configuration commands on just one node within the HA group. Alternatively, if you prefer, you can manually copy the commands to each individual node as an alternative approach.
- The HA group operates as a Pacemaker cluster, and configuring this Pacemaker cluster involves several steps. Initially, you define the Pacemaker cluster by making adjustments to the `/var/mqm/rdqm.ini` file. In this particular configuration, each interface is allocated a distinct IP address, as demonstrated in Example 7-5.

*Example 7-5 Interface configuration*

```
Node:
  Name=node-a.powerm.ma
  HA_Primary=192.168.100.1
  HA_Alternate=192.168.101.1
  HA_Replication=192.168.102.1
```

```
Node:
  Name=node-b.powerm.ma
  HA_Primary=192.168.100.2
  HA_Alternate=192.168.101.2
  HA_Replication=192.168.102.2
Node:
  Name=node-c.powerm.ma
  HA_Primary=192.168.100.3
  HA_Alternate=192.168.101.3
  HA_Replication=192.168.102.3
```

Once the configuration is complete, follow these steps:

- Edit the `/var/mqm/rdqm.ini` file on one of the three servers to define the cluster.
- Duplicate the modified file onto the remaining two servers that will serve as nodes within the Pacemaker cluster.
- Execute the command shown in Example 7-6 as the root user on each of the three servers:

*Example 7-6 rdqmadm command*

```
$ rdqmadm -c
```

- Next, create a Replicated Data Queue Manager with Root Privilege by executing the command in Example 7-7 on the secondary nodes (Node B and Node C). This creates the secondary queue manager RDQM1.

*Example 7-7 Create a secondary queue manager on nodes Node B and Node C*

```
$ ./crtmqm –sxs –fs 3072M RDQM1
Creating replicated data queue manager configuration.
IBM MQ secondary queue manager created.
```

- On the primary node (Node A), use the command shown in Example 7-8 to create the primary queue manager RDQM1.

*Example 7-8 Create a primary queue manager*

```
$ ./crtmqm –sx –fs 3072M RDQM1
Creating replicated data queue manager configuration.
IBM MQ queue manager created.
Directory '/var/mqm/vols/rdqm1/qmgr/rqdm1’ created.
The queue manager is associated with installation 'Installation1'.
Creating or replacing default objects for queue manager 'RDQM1'.
Default objects statistics : 84 created. 0 replaced. 0 failed.
Completing setup.
Setup completed.
Enabling replicated data queue manager.
Replicated data queue manager enabled.
```

- To retrieve the status of the RDQM, use the command shown in Example 7-9.

*Example 7-9 Checking the status of RDQM*

```
$ rdqmstatus -m RDQM1
Node: node-a.powerm.ma
Queue manager status: Running
CPU: 0.07%
Memory: 267MB
```

```
Queue manager file system: 67MB used, 2.9GB allocated [2%]
HA role: Primary
HA status: Normal
HA control: Enabled
HA current location: This node
HA preferred location: This node
(…)

Node: node-b.powerm.ma
HA status: Normal

Node:  node-c.powerm.ma
HA status: Normal
```

## 7.2 Using VM Recovery Manager to Replicate Pacemaker cluster

This section provides an overview on how to configuration VM Recovery Manager DR to provide recovery of a Pacemaker cluster in a second data center in order to provide a disaster recovery solution.

The production Pacemaker cluster is running in the primary data center and Pacemaker is providing fail-over support for nodes within the cluster to provide HA. However, if the primary data center is faced by a critical failure or disaster that takes the whole data center off line, then the workloads need to be recovered in a secondary data center which is not affected by the disaster. One solution to this issue can be provided by using IBM VM Recovery Manager Disaster Recovery (VMRM DR).

VMRM DR is an IBM solution that utilizes storage replication and virtual machine restart technologies to restart all of the Pacemaker cluster nodes to a known state in a controlled fashion in your backup site.

### 7.2.1 VM Recovery Manager DR Overview and Architecture

The ability to recover applications and services is a key component in providing continuous business services. The VMRM DR for IBM Power solution is a DR solution that is easy to deploy and provides automated operations to recover the production site. The VMRM DR solution is based on the IBM Geographically Dispersed Parallel Sysplex® (IBM GDPS) offering concept that optimizes the usage of resources. This solution does not require you to deploy backup VMs for DR. Thus, the VMRM DR solution reduces the software license and administrative costs.

Unlike the clustered-based technology that uses redundant hardware and software components to provide fail over in a component failure, VMRM DR is developed by using VM restart-based technology, where an out-of-band monitoring and management component is used to restart the VMs when a failure of the host infrastructure occurs.

Figure 7-3 on page 133 shows an overview of VMRM DR. IBM VMRM DR was formerly known as IBM Geographically Dispersed Resiliency (IBM GDR) and provides a disaster recovery (DR) solution using virtual machine restart technology.

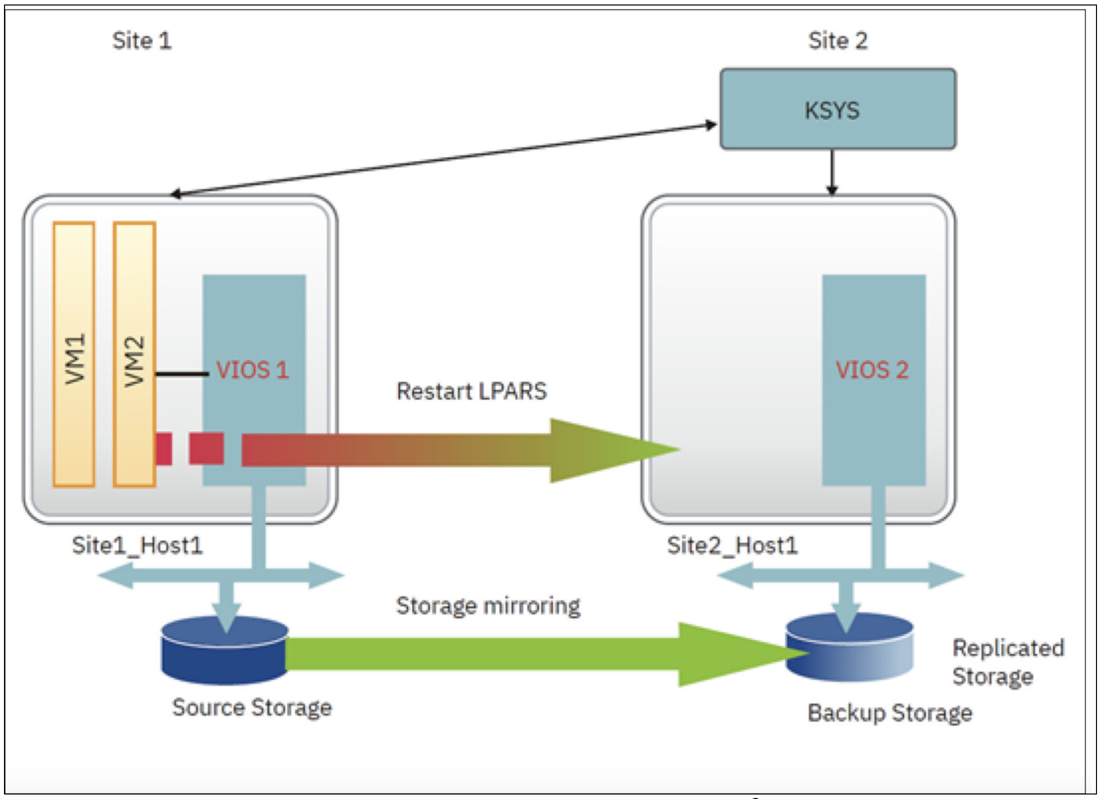


*Figure 7-3 Disaster recovery Architecture for VM Recovery Manager.*[3]

In this architecture VM1 and VM2 can be assumed to be members of your Pacemaker cluster. In case all the nodes of Pacemaker cluster are failed, it can be restarted on the backup site with replicated storage. VMRM-DR is a DR solution that is easy to deploy and provides automated operations to recover the production workloads at the remote site.

VMRM DR is based on the IBM Geographically Dispersed Parallel Sysplex (IBM GDPS®) offering that optimizes the usage of resources. This solution does not require you to deploy the backup VMs for DR. Thus, the VMRM DR solution reduces the software license and administrative costs.

The VM restart-based HADR solution relies on an out-of-band monitoring and management component that restarts the VMs on other hardware when the host infrastructure fails which is shown in Figure 7-3 as the KSYS. This solution provides an easy deployment model that uses the KSYS to monitor the entire VM environment and take the appropriate recovery action based on user defined and flexible failover policies along with management of the storage replication.

Table 7-1 on page 134 identifies the differences between a conventional cluster-based DR model and the VMRM DR solution.

[3] Overview for IBM VM Recovery Manager DR for Power Systems

*Table 7-1 Clustered DR versus VM Recovery Manager DR*

| **Parameters** | **Cluster-based DR model** | **VM restart DR model that is used by the VM Recovery Manager DR solution** |
|---|---|---|
| Deployment method | Redundant hardware and software components are deployed at the beginning of the implementation to provide near real-time failovers when some of the components fail. | With virtualization technology, many images of the OS are deployed in a system. These VMs are deployed on physical hardware by the hypervisor that allocates and manages the CPU, memory, and I/O physical resources that are shared among the VMs. |
| Dependency | This solution relies on the monitoring and heartbeat capabilities within the cluster to monitor the health of the cluster and take recovery action if a failure condition is detected. | This solution relies on out-of-band monitoring software that works closely with the hypervisor to monitor the VM environment and to provide a DR mechanism for the VM environment. |
| Workload startup time | The workload startup time is faster because the VMs and the software stack are already available. | The VMs require more time to restart in the backup environment. |
| Cluster administration required | Yes | No |
| Error coverage | Comprehensive. This solution monitors the entire cluster for any errors. | Limited. This solution monitors the servers and the VMs for errors. |
| Deployment simplicity | This solution must be set up in each VM. | Aggregated deployment at the site level. |
| Protected workload type | Critical workloads can be protected by using this solution. | Critical workloads can be protected by using this solution. |
| Software license and administrative costs | This solution costs more because redundant software and hardware are required to deploy this solution. | This solution costs less because of optimized usage of resources. |

**Demonstration:** A demonstration of VMRM DR under its original name of IBM GDR is available at IBM Geographically Dispersed Resiliency (GDR) for IBM Power Systems Demo.

VMRM DR has the added feature of supporting DR testing. A copy of the storage at the remote data center is copied and zoned to the DR test VMs, which then can be started in a sandbox environment for testing.

### 7.2.2 Installation and Configuration requirements

To perform an installation task, root authority is required.

The KSYS LPAR must have at least one core CPU and 8 GB of memory. These requirements can be higher if you have a large environment of more than 100 LPARs in the data center.

As a best practice, you must deploy AIX rules in the VIOS. The VIOS must have enough available space in the / (root), /var, and /usr file systems. Extra CPU and memory resources are needed in each VIOS for VMRM HA management.

The VIOS must have enough available space in the / (root), /var, and /usr file systems and add at least 1-core CPU and 4 GB memory to the sizing that you are planning to deploy on your production environment, and have at least 1-core CPU and 10 GB memory in a scalable environment.

Ensure that you have enough space in the LPAR so that KSYS file sets can be installed successfully. You must have 30 MB of disk space in the /opt directory and 200 MB of disk space in the /var directory. Check whether a KSYS installation is already in progress by using the **ksysmgr q cl** command. If the KSYS software was installed previously, you must uninstall the KSYS software.

For planning requirement and installation requirement for VM Recovery Manager DR solution implementation, see Planning VM Recovery Manager DR.

## 7.2.3 Disaster recovery mechanism for the VM recovery manager DR solution

After you plan the details about how the VM Recovery Manager DR solution can integrated into your current environment, review the following flow chart that contains the high-level steps that are involved in the VMRM DR implementation.

Figure 7-4 provides a summary of the entire VMRM DR mechanism for DR:

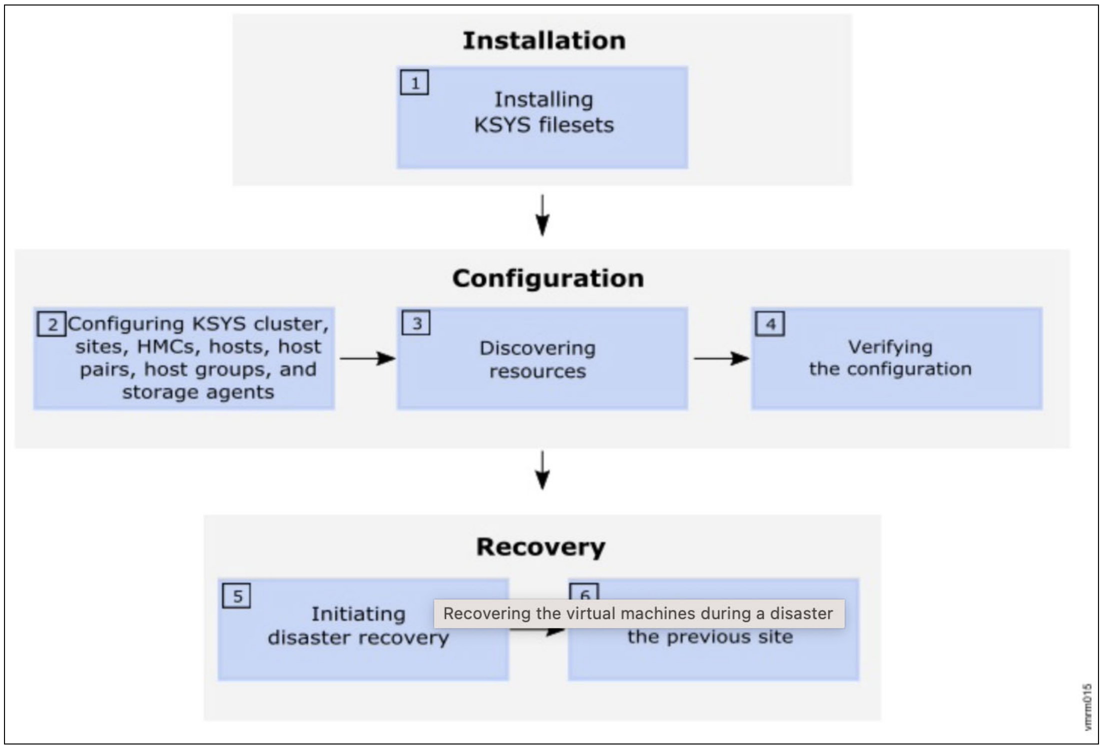

Installation
1 Installing KSYS filesets
Configuration
2 Configuring KSYS cluster, sites, HMCs, hosts, host pairs, host groups, and storage agents
3 Discovering resources
4 Verifying the configuration
Recovery
5 Initiating disaster recovery
6 the previous site
Recovering the virtual machines during a disaster
vmrm015

*Figure 7-4 VM Recovery manager DR solution: Disaster recovery mechanism*

The boxes in the flow chart are described as:

1. Installation

   The controlling system (KSYS) is the fundamental component in the VM Recovery Manager DR solution. Therefore, the KSYS filesets must be installed first. The KSYS runs in an AIX 7.2.1 (or later) logical partition in the disaster recovery site. It controls the entire cloud environment for the VM Recovery Manager DR solution. To manage the servers and data replication, the KSYS must be connected to all the managed servers through the HMC and out-of-band storage subsystem connectivity to all associated primary and secondary disks.

   Refer to this VMRM DR document for details on the installation of VM Recovery manager DR solution on controller node.

2. Configuration

   After the KSYS is installed and a one-node KSYS cluster is set up, you must configure all the other entities by using the KSYS interface. You must complete the following procedures by using the **ksysmgr** command:

   The details on the ksysmgr command including syntax can be found in this VMRM document.

   – Create a one-node cluster for the KSYS node.
   – Create sites.
   – Add HMCs to the corresponding sites.
   – Add hosts to the corresponding sites.
   – Identify host pairs across the sites.
   – Create host groups and work groups.
   – Add storage agents to the corresponding sites.
   – Add contacts details for error notification.

   Refer to this KSYS configuration document for configuring KSYS subsystem.

3. Discovering the resources.

   After the initial configuration is complete, the KSYS discovers all the hosts from all the host groups that are managed by the HMCs in both sites and displays the status.

   During discovery, the KSYS subsystem monitors the discovery of all logical partitions (LPARs) or virtual machines (VMs) in all the managed hosts in the active site. The KSYS collects the configuration information for each LPAR, and displays the status, and also logs the status in the log files at the/var/ksys/log/directory.

   The KSYS discovers the disks of each VM and checks whether the VMs are configured currently for the storage devices mirroring. If the disks are not configured for mirroring properly, KSYS notifies you about the volumes that are not mirrored. All volumes of a VM must be mirrored. Disks can be virtualized by using N_Port ID virtualization (NPIV), virtual SCSI (vSCSI), or a combination of all these modes.

   The KSYS subsystem collects information about the hosts, VIOS, and logical partitions that can be managed from the HMC during the discovery phase. For example, the KSYS subsystem collects information about the system processor, system memory, hardware, and worldwide port name (WWPN) of the physical Fibre Channel adapter. The KSYS subsystem also checks for the VIOS capability for disaster recovery operations. The KSYS subsystem also collects the dynamic information about the host state, LPAR state, VIOS state, and IP addresses of the host, VIOS, and LPAR.

   Command to trigger discover:

   ```
   ksysmgr -t discover site <source_site>.
   ```

   Discovery can be triggered at site level, host group level or workgroup level.

**Note:** After a change in the configuration, running discovery is mandatory to reflect the changes in configuration.

Refer to this VMRM Document for more details on VM Recovery manager DR discovery operation.

4. Verification of the resource for DR readiness

   In addition to the configuration validation that you initiated, the KSYS verifies and validates the environment periodically. The KSYS also verifies the configuration as part of the recovery process. In the verification phase, the KSYS fetches information from the HMC to check whether the backup site is capable to host the VMs during a disaster. The KSYS also verifies storage replication-related details and accessibility of the target disks. The verification is successful only if the storage area network (SAN) zones are configured properly on the target side.

   If the verification fails as a part of the recovery process, the failure is considered as recovery failure.

   Command to execute verification:

   ```
   ksysmgr -t verify site <Source site>.
   ```

   **Note:** Verification can be triggered at site level, host group level and at workgroup level.

   Refer to this VMRM document for more details on verification operations.

5. Recovery of the resource in case of production failure

   When any planned or unplanned outages occur, you must manually initiate the recovery by using the **ksysmgr** command that moves the virtual machines to the backup site. If you initiate a planned recovery, the storage replication direction is reversed from the current active site to the previously active site. If you initiate an unplanned recovery, the storage is failed over to the backup site and you must manually resynchronize the storage after the previously active site becomes operational.

   For planned recovery trigger command:

   ```
   ksysmgr -t move site from<source_site> to=<backup_site>
   ```

   For unplanned disaster recovery failure triggered command:

   ```
   ksysmgr -t move site from=<source_site> to=<backup_site> dr_type=unplanned
   ```

   **Note:** The move command can be triggered at site level, host group level or workgroup level.

   For more information, see Recovering the virtual machines during a disaster.

6. Cleanup of resource after DR movement.

   After the disaster recovery phase, when the virtual machines have been moved to the target site, you must clean up the source site. The site cleanup process is different for planned recovery operation and unplanned recovery operation. In case of a planned recovery, the KSYS automatically cleans up the source site of all the disk mapping and adapter information. In the case of an unplanned recovery, the source site components such as HMC and storage subsystems might not be operational. So, you must manually clean up the source site when the HMC and hosts in the previously active site have become operational.

The latest release of VM recovery manager is 1.7. Refer to this product guide for more information.

### Conclusion

By using VM Recovery manager and registering Power system virtual machines having a Pacemaker cluster, can be monitored by KSYS controller node. On disaster or critical hardware failure, Pacemaker cluster nodes on the target site can be recovery using replication disk on target site having Pacemaker cluster node details.

## 7.3 NSF Server HA

Configuring NFS Server HA using Pacemaker involves several steps to ensure a robust and failsafe setup. This setup aims to maintain continuous access to NFS shares even if one server fails. Below are the general steps involved in this configuration:

- Begin by installing Pacemaker on each server that will be part of the HA setup. Configure these tools to facilitate communication and coordination between the servers.
- Set up the NFS server on each node, ensuring that the NFS shares are correctly configured and accessible by the clients. Verify that the NFS service is working as expected before proceeding further.
- Create Pacemaker resource agents for the NFS server. These agents will manage the NFS services, monitor their status, and control their behavior in case of failures. Using Pacemaker, define the NFS server as a cluster resource. This includes specifying the NFS service, IP addresses, and any other necessary resources that need to be managed in an HA scenario.
- Establish constraints and failover policies within Pacemaker to control how the resources should behave during various failure scenarios. Define rules for resource placement and recovery in case of node failures or network issues.
- Simulate failures, node reboots, and network interruptions to ensure that Pacemaker responds correctly and fails over the NFS service seamlessly without disrupting client access.
- Implement monitoring tools or scripts to continuously monitor the health and performance of the NFS servers. Regularly maintain and update the HA configuration as needed.

Before configuring NFS Server HA using Pacemaker, certain prerequisites must be fulfilled. These include:

- Red Hat Enterprise Linux 8 and 9: Ensure that the servers intended for NFS Server HA are running Red Hat Enterprise Linux (RHEL) versions 8 or 9. Verify compatibility and adherence to the specific requirements for Pacemaker and associated tools on these RHEL versions.
- Enable the HA or Resilient Storage Add-Ons for RHEL on each server participating in the HA cluster. These add-ons provide the necessary tools and features to establish a robust, fail-safe environment using the commands shown in Example 7-10 and Example 7-11 on page 139.

*Example 7-10 Enable Red Hat Hight Availability Add-On Repository*

```
$ sudo subscription-manager repos --enable=rhel-9-for-ppc64le-highavailability-rpms
Repository 'rhel-9-for-ppc64le-highavailability-rpms' is enabled for this system.
```

*Example 7-11 Enable Red Hat Resilient Storage Add-On Repository*

```
$ sudo subscription-manager repos --enable=rhel-9-for-ppc64le-resilientstorage-rpms
Repository 'rhel-9-for-ppc64le-resilientstorage-rpms' is enabled for this system.
```

**Note:** The Red Hat Enterprise Linux Resilient Storage Add-On enables simultaneous shared storage access within a highly available cluster. Each server within the cluster necessitates direct access to a shared block device, achievable through a local storage area network (SAN) or provided by various public cloud providers. This Add-On encompasses:

- The Global File System 2.
- The Red Hat Enterprise Linux HA Add-On.
- A clustered volume manager, with RHEL 8.4 and higher versions supporting encrypted volumes. An active/active clustered Samba (SMB/CIFS) server.
- Performance Co-Pilot.

- Set up an Active/Passive configuration for shared storage among the nodes in the cluster. This configuration ensures that only one node (the active node) manages the shared storage while the others (passive nodes) remain on standby, ready to take over in case of a failure. Verify that the shared storage is accessible by all nodes and can be seamlessly switched between active and passive modes without compromising data integrity.

The subsequent links provide detailed instructions for setting up NFS Server HA using Pacemaker, applicable to either Linux on IBM Power or Linux on IBM Power Virtual Server.

- Configuring an Active-Passive NFS Server in a Red Hat High Availability Cluster
- How to configure HA-LVM Cluster using system_id in RHEL 8 and above?
- Configuring and managing high availability clusters

## 7.4 IBM MQ active/passive architecture using Pacemaker and NFS on IBM Power Virtual Server

To configure IBM MQ HA in an active-passive configuration, you can use the Red Hat Enterprise Linux HA Add-On. This setup requires the following components:

- IBM Power Virtual Server with a Linux Operating System deployed in either a single or multiple IBM Cloud locations.
- Allocation of storage to the Logical Partitions (LPARs) within the IBM Power Virtual Server environment.
- Implementation of an IBM network linking IBM Cloud locations, especially if multiple data centers are selected. The HA repository is correctly enabled.
- Deployment of IBM MQ version 9.2 or above

For comprehensive guidance and detailed instructions, see *IBM Power Systems High Availability and Disaster Recovery Updates: Planning for a Multicloud Environment*, REDP-5663.

## 7.5 IBM PowerVC three-node architecture using Pacemaker

IBM PowerVC (Power Virtualization Center) is a software solution designed to simplify the management and deployment of virtualized resources on IBM Power Systems servers. It serves as a comprehensive tool for managing virtual environments, offering capabilities for provisioning, monitoring, and optimizing virtualized infrastructure. Key features of IBM PowerVC include:

- PowerVC enables users to create, deploy, and manage virtual machines (VMs) on IBM Power Systems servers. It provides a user-friendly interface for VM creation, resource allocation, and configuration.
- It allows administrators to pool and efficiently manage physical resources such as CPU, memory, storage, and network, optimizing their utilization across multiple virtual machines.
- PowerVC offers automation capabilities for streamlining routine tasks, such as VM provisioning and resource allocation. It also supports self-service portals, enabling users to request and deploy VMs within defined resource constraints.
- It integrates with various virtualization technologies, including PowerVM, IBM hypervisor technology for Power Systems. PowerVC supports multiple operating systems and is designed to work seamlessly with Power Systems architecture.
- It provides monitoring tools to track the performance and health of virtualized environments. It offers reporting functionalities for resource usage, allowing administrators to optimize resource allocation.

PowerVC 2.0.2 and higher offer support for a three-node architecture. The multi-node setup encompasses specific elements aimed at delivering an HA solution, outlined in subsequent sections here. In this deployment model, the majority of services operate in an Active-Active mode, overseen by Pacemaker or Corosync for monitoring.

For a comprehensive, step-by-step deployment approach, please consult the following publication and link:

- *IBM Power Systems High Availability and Disaster Recovery Updates: Planning for a Multicloud Environment*, REDP-5663.
- PowerVC high availability and scale architecture

# 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

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.

- *High Availability and Disaster Recovery Options for IBM Power Cloud and On-Premises*, REDP-5656
- *IBM Power Systems High Availability and Disaster Recovery Updates: Planning for a Multicloud Environment*, REDP-5663
- *IBM Virtual Machine Recovery Manager for IBM Power Cookbook*, SG24-8539
- *SAP HANA on IBM Power Systems Virtual Servers: Hybrid Cloud Solution*, REDP-5693

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

**ibm.com**/redbooks

## Online resources

These websites are also relevant as further information sources:

- Automating SAP HANA Scale-Up System Replication using the RHEL HA Add-On (RHEL8)
- Automating SAP HANA Scale-Up System Replication using the RHEL HA Add-On (RHEL9)
- Cluster Labs Pacemaker Documentation
- DB2 High Availability strategies
- DB2 High availability disaster recovery (HADR)
- High availability with Db2 server
- Red Hat Enterprise Linux Configuring and managing high availability clusters: Using the Red Hat High Availability Add-On to create and maintain Pacemaker clusters
- Red Hat HA Solutions for SAP HANA, S/4HANA and NetWeaver based SAP Applications

## Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

Redbooks

Using Pacemaker to Create Highly Available Linux Solutions

SG24-8557-00

ISBN 0738461733

IBM

(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

Redbooks

Using Pacemaker to Create Highly Available Linux Solutions on IBM

SG24-8557-00

ISBN 0738461733

IBM

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Redbooks

Using Pacemaker to Create Highly Available Linux Solutions on

SG24-8557-00

ISBN 0738461733

IBM

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

Redbooks

Using Pacemaker to Create Highly Available Linux Solutions on IBM Power

(0.2”spine)
0.17”<->0.473”
90<->249 pages

(0.1”spine)
0.1”<->0.169”
53<->89 pages

Redbooks

Using Pacemaker to Create Highly Available

SG24-8557-00

ISBN 0738461733

IBM

(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages

Redbooks

Using Pacemaker to Create Highly Available Linux Solutions on IBM Power

SG24-8557-00

ISBN 0738461733

IBM

(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

SG24-8557-00

ISBN 0738461733