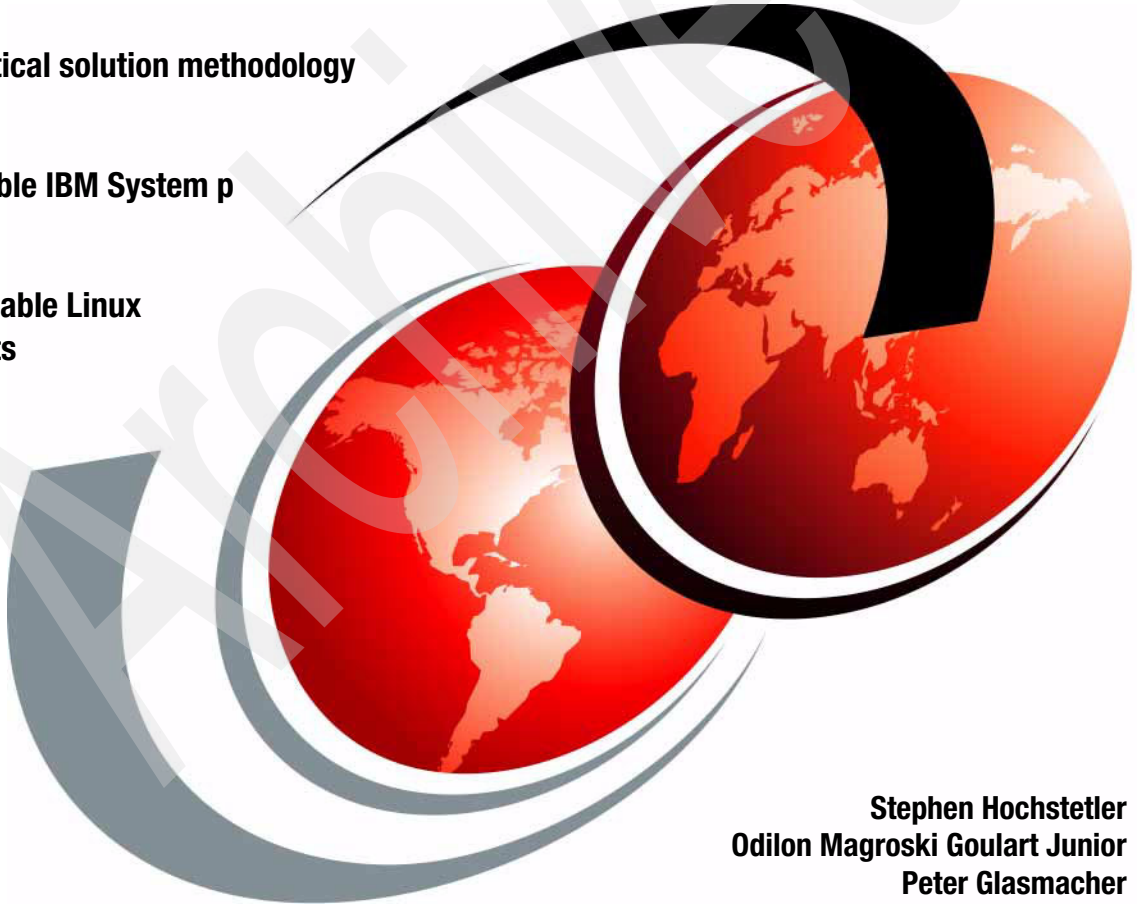
IBM

Deploying Mission Critical Applications with Linux on POWER

- Mission critical solution methodology
- Highly reliable IBM System p hardware
- Highly available Linux deployments



Stephen Hochstetler
Odilon Magroski Goulart Junior
Peter Glasmacher

Redbooks

ibm.com/redbooks

International Technical Support Organization

Deploying Mission Critical Applications with Linux on POWER

October 2007

SG24-7286-00

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

First Edition (October 2007)

This edition applies to System p hardware and the Linux OS.

Contents

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

AIX 5L™	i5/OS®	Redbooks (logo) ®
AIX®	IBM®	RS/6000®
BladeCenter®	OpenPower™	ServeRAID™
Chipkill™	PowerPC®	System i™
DB2 Universal Database™	POWER™	System p™
DB2®	POWER4™	System p5™
Enterprise Storage Server®	POWER5™	Tivoli®
eServer™	POWER5+™	TotalStorage®
FlashCopy®	pSeries®	Virtualization Engine™
HACMP™	Redbooks®	WebSphere®

The following terms are trademarks of other companies:

ABAP, mySAP, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

IPX, Java, J2EE, Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

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

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

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

SAP NetWeaver, SAP Solution Manager and mysap are the trademarks or registered trademarks of SAP AG in Germany and in several other countries.

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

Preface

This IBM® Redbooks® publication gives you the information you need to help you run mission critical applications in an IBM System p™ environment. Matching applications, such as SAP®, and Linux® highly available solutions with IBM System p systems provides you with the capabilities to produce a highly available solution.

This book will help you think through the process of setting up new solutions or help you review an existing solution. If you are on an previous generation IBM POWER™ server or running Linux on another architecture, this book will help you understand the benefits of moving your critical applications to Linux running on IBM System p servers.

The team that wrote this IBM Redbooks publication

This IBM Redbooks publication was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Stephen Hochstetler is an IT Specialist at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide on all areas of Linux, System p, and Tivoli® products. Before joining the ITSO six years ago, Stephen worked in Tivoli Services, USA as a network management specialist.

Odilon Magroski Goulart Junior is an SAP Consultant in ITS Brazil. He has five years of experience in planning, implementing, and supporting SAP solutions. His areas of expertise includes network, database, operational, and SAP systems. He holds some certifications in these areas.

Peter Glasmacher is a certified IBM expert professional from Dortmund, Germany. After receiving a degree in communication electronics, he joined IBM in 1973. He has worked in various positions, including support, development, and services, covering multiple operating system platforms and architectures. Currently, he works as a consulting IT specialist for IBM Global Technology Services. Since 1983, he has written extensively on workstation-related issues. He has co-authored several IBM Redbooks publications, covering network and systems management topics.

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

Alan Robertson
Ken Rosendal
IBM Linux Technology Center

Marcelo Comitre, Strategic Outsourcing, PeopleSoft® Specialist, Hortolandia
IBM Brazil

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbooks publication dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will team with IBM technical professionals, Business Partners, and customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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

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

  ibm.com/redbooks
- Send your comments in an e-mail to:

  redbook@us.ibm.com
- Mail your comments to:

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

1

Introduction

In this introductory chapter, we provide some background information about Linux, IBM System p servers, and other topics covered throughout this book.

First, we discuss what makes an application mission critical. We then present an introduction to the POWER architecture and explain the reliability of this platform. We then review the architecture of the Linux operating system (OS), describe the design points of it, and explore the benefits of running Linux OS. There are many Linux distributions throughout the world, so it is important for you to know which distributions take advantage of the features of the System p servers. The last topic discussed in this chapter is the deployment of Linux in an enterprise environment.

Linux is a general purpose operating system that provides all the required components needed to run it on a desktop or a server platform and execute mission critical applications.

Linux on POWER is a term used when running Linux on the family of POWER and PowerPC® processors that are available in a variety of products that range from embedded processors for cellphones to processors for the largest supercomputers.

Linux on POWER is a common term for running Linux on a number of product lines, such as:

- IBM System p servers
- IBM System i™ servers
- OpenPower™ servers
- BladeCenter® JS20 and JS21 blades

All these product lines are based on the IBM POWER architecture. They are enabled to run Linux distributions compiled for the POWER platform, such as distributions from Novell SUSE or Red Hat. For more information, visit the following Web site:

http://www.ibm.com/systems/p/linux/

At the time of writing, more than 3200 applications for large and small businesses running under Linux on POWER have been completed.

For more discussions on Linux on POWER, visit the following Web site:

http://oss.gonicus.de/openpower/index.php/Main_Page

which hosts a wiki on Linux on Power. Also, there is a IBM wiki that can be found at the following Web site:

http://www.ibm.com/collaboration/wiki/display/LinuxP/Home

Actual information about IBM efforts in the Linux arena can be found at the following Web site:

http://www.ibm.com/developerworks/linux/

Specific information and supplemental material about Linux on POWER can be found at the following Web site:

http://www.ibm.com/developerworks/linux/power/

1.1 What makes an application mission critical

Mission critical is an expression or IT term used to symbolize the importance of the application or environment to the customer. When an environment is considered to be mission critical by the customer, they want to do everything possible to keep this environment constantly available. If something happens to this environment, the customer will be impacted. Therefore, this environment must be built to avoid temporary outages, no matter how minor they are, and to reduce the possibility of data loss. Usually, environments considered mission critical need to maintain an annual readiness of 99.89%. For example, if the year has 8766 hours, the maximum downtime during the year will be 10 hours, excluding scheduled downtimes.

To maintain an mission critical environment with this level of readiness and not compromise the Service Level of Agreement (SLA), we need to use the highest level of fault tolerance on each item in this environment, using hardware redundancy, software configuration, two or more network paths, RAID storage, double power supplies and fans, standby servers, High Availability Solutions, and virtual servers.

These hardware and software technologies will increase the Total Cost of Ownership (TCO) of the solution, because they will need hardware and software to prepared this mission critical environment for any kind of possible problems. Therefore, the main decision of whether an environment should be mission critical will be also be influenced by the financial investment in technology and cost of maintenance of the environment. Choosing to make this investment is a risk mitigation decision.

1.2 An introduction to POWER

In this section, we provide a brief overview of the actual POWER5™ and POWER5+™ architecture. For actual hardware and packaging options, please go to the following Web site:

http://www.ibm.com/systems/p/hardware/

The IBM System p server, IBM System i server, OpenPower server, and BladeCenter JS20 and JS21 blades are based on POWER5 and POWER5+ architectures. POWER5 and POWER5+ are the current generations of POWER processors.

A POWER5 server is scalable up to a 64-way symmetric multiprocessor configuration. POWER5 comes with two processor cores on a single chip.

The POWER5+ servers use 90 nanometer technology. This results in a reasonable size reduction and higher bus rates. In addition, POWER5+ servers provide more packaging configurations:

- Dual-core, featuring two 64-bit processor cores on a chip
- Quad-core, featuring four 64-bit processors on a chip
- Multi-chip, featuring eight 64-bit processors on a chip

Figure 1-1 shows the packaging. Note that each processor pair has its own level3 cache.

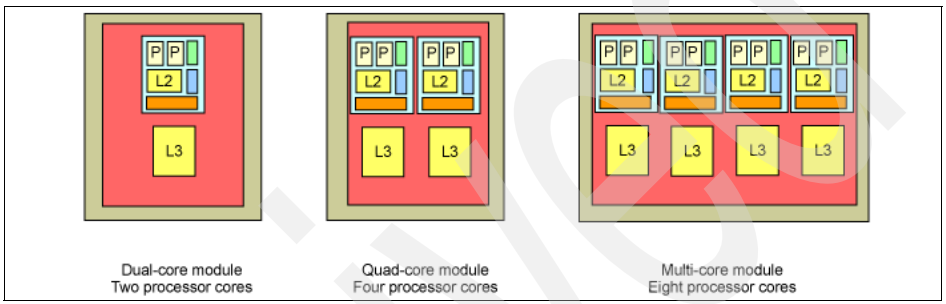


Figure 1-1 POWER5+ packaging options

Besides the actual packaging of multiple processors on a single chip, POWER5 has a number of additional features pertinent to the architecture:

- Simultaneous multithreading
- A virtualization engine
- The RAS (Reliability, Availability, Serviceability) technology found in mainframe systems

Simultaneous multithreading

This feature allows a POWER5 core to execute two sequences of instructions at the same time.

Virtualization engine

This feature is used to assign storage, disk, network, and processor resources in a single virtual environment. Through virtualization, the execution of multiple operating systems on a single machine is made possible.

RAS

This feature includes a number of hardware and software mechanisms ranging from dedicated service processors to extended error checking. They are as follows:

Service processor A separate microprocessor used for surveillance and error log functions. In addition, the service processor monitors a special firmware heartbeat and cycles the machine in case the heartbeat is missing.

First Failure Data Capture (FFDC)
This feature can be used by the software to record failures and software incidents. It also helps find the cause for a problem and supports root cause analysis.

Chipkill™ memory This feature allows a motherboard to detect memory problems and selectively disable parts of the memory.

Memory error checking (ECC)
The memory cards used with POWER5 have ECC circuits that detect memory double errors and are able to correct single bit errors.

Dynamic processor deallocation and logical partition error containment
If a processor exceeds a threshold of recoverable errors, such as on an L2 cache access, the event is logged. In addition to logging the error, the processor will actually be marked and deconfigured from the system while the operating system continues to run. This feature allows the processor to be repaired on a deferred basis while helping prevent an unscheduled system outage.

Dynamic firmware maintenance
This feature allows you to update the firmware and apply fixes without disrupting the system.

Hot I/O drawer Hot I/O drawer supports the installation of remote I/O drawers without disrupting the system.

1.3 Introduction to Linux

In this section, we like to review the history and the current status of Linux as an operating system and a development environment.

We show the evolution of Linux and discuss its increasing importance as a general purpose operating system.

1.3.1 A brief history of Linux

Linux is a free, UNIX®-like operating system, but it is NOT UNIX. Its first working sample was published in 1991. At that time, commercial IT systems, whether they were so-called mainframe systems or smaller distributed systems, were built on proprietary operating systems, that is, you had to use the operating system that was developed and delivered together with the hardware. In this context, even UNIX and its derivatives were considered proprietary operating systems.

At the same time, the Personal Computer, or PC, introduced some 10 years ago, became a significant part of the IT world. However, these relatively cheap computers for everybody had the same drawback: They were hosting one of a few available proprietary operating systems, and applications and development tools were proprietary as well.

In midst of this situation, Richard Stallman founded the Free Software Foundation (FSF), which had the goal of developing software whose source code was freely available. Stallman then introduced another milestone known as the GNU project. The GNU project was founded to develop a freely available UNIX-like operating system. In the late 1980s and early 1990s, UNIX (and its derivatives) had a good reputation as a resource-aware operating system. In many IT environments, UNIX was the operating system of choice for smaller distributed computers. Today, we would call those machines *servers*.

In 1991, the GNU project developed almost any essential UNIX tool, including the C compiler *gcc*. What was missing was an operating system kernel and file system support. That year, a young Finnish student named Linus Torvalds developed an operating system kernel inspired by Minix, a simple but real operating system. Minix was developed for educational purposes by Andrew S. Tanenbaum, a Dutch professor, and ran on an Intel® 8086 processor, as detailed in *Operating Systems Design & Implementation* by Tanenbaum.

Torvalds used Minix for his studies. Unfortunately, Tanenbaum did not allow Minix to be modified or extended, so Torvalds wrote a complete new multithreading capable operating system kernel and put it under the GNU license and named it Linux.

The early versions of Linux required another operating system to boot the system and they lacked many essential components. But the kernel worked and the GNU compiler gcc executed under Linux. After a short time, a large number of enthusiasts were developing additions to the operating system kernel and enabled the existing GNU utilities and programs to run under Linux.

Later on, a flexible and versatile boot loader was added and the module subsystem was introduced, which enabled the addition and removal of kernel functions such as new hardware drivers while the system was running.

Over the next five years, Linux became a stable and mature operating system, featuring all the things required for both being deployed on a server or on a desktop.

Today, Linux is an full fledged operating system which offers all the required services for serious IT operations. Its open architecture allows the fast implementation of new components, such as new hardware.

Linux is a UNIX like operating system, which means that most common UNIX utilities and programs execute under Linux. Linux was developed to run on an Intel 80386 platform and used its specific interrupt system, so it was not portable at all. Because the source code of the kernel was and is still freely available, efforts to port Linux to other platforms started early in the lifetime of Linux.

Today, Linux runs on a number of different platforms. POWER, the platform we will focus in this book, is just one of those platforms.

Over the years, Linux has evolved to a mature POSIX-like operating system. It provides

- True multitasking
- A flexible memory management, including virtual memory management
- An on demand loading feature that allows you to dynamically load and unload kernel functions and device drivers
- A built in TCP/IP network stack
- A number of other features common to UNIX operating systems

1.3.2 Architecture overview

Linux, like many other operating systems, is a collection of programs running on a particular hardware platform. The operating system manages the available resources and allow applications to be executed on top of the operating system.

To achieve these goals, Linux was built as an monolithic kernel providing essential system services, as shown in Figure 1-2. Basically, a program executes under Linux inside a process. As a true multitasking system, Linux is able to manage multiple processes at the same time.

However, only one process is actually dispatched to one signal processor core and is allowed to execute. To overcome this limitation, Linux supports symmetric multiprocessing (SMP). In other words, Linux can control hardware environments featuring multiple processors.

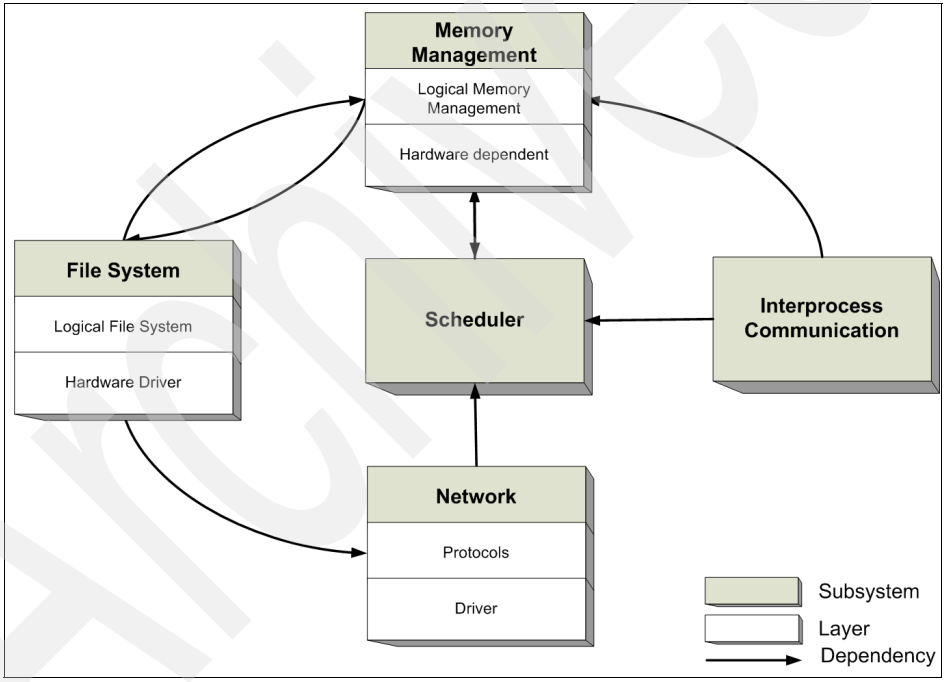


Figure 1-2 The building blocks of the Linux kernel

In addition to a CPU, which executes code, a process or an application in a more general view needs a number of other resources in order to run an application. Thus, a Linux kernel, on a high level, is built out of five different blocks:

- A scheduler, which provides basic multitasking services. The scheduler manages all currently loaded processes and dispatches them to the CPU.

- A memory manager, which manages the physical memory available and provides paging capabilities and maps virtual addresses accessed by a process to physical addresses. The memory management subsystem consists of a hardware dependent part and a hardware independent part. This allows Linux to be implemented on various hardware platforms.
- A file system in order to store and retrieve data in the form of files. Linux offers a number of different file systems and a logical volume manager subsystem.
- A network subsystem. Linux implements a socket based networking subsystem build on INET and BSD type sockets. On top of the socket, communication protocols stacks, such as TCP/IP or IPX™, can be implemented
- An interprocess communication scheme. Linux allows processes to share available resources and exchange data between processes. It provides a versatile set of mechanisms for the communication between processes. Signals are the standard UNIX style of an interprocess communication. Linux supports signal as well as more versatile mechanisms like the so-called System V IPC, shared memory, or communication between processes through pipes and named pipes.
- Last but not least, Linux provides an user interface. The user interface is not an integral part of the kernel, but its importance requires mention. Originally, the Linux user interface was command line oriented, which is commonly known as the shell. Linus Torvalds implemented the bash (Bourne again shell) in his original system. Even today, bash is the command interface of choice for many Linux users. However, a number of graphical interfaces are available or are being distributed with Linux. Popular graphical environments such as KDE or GNOME have contributed to Linux's increasing presence on desktop computers.

As you can further see in Figure 1-2 on page 8, various components of the kernel are built from multiple blocks, in other words, Linux features a layered design, which makes the operating system very flexible.

Because of its layered design, a modern Linux kernel is no longer a monolithic piece of code. Today's Linux provides a relative small kernel that is sufficient to initialize itself and then dynamically load all the additional resources in the form of modules at runtime. Having the code implement various services in a modular format, that is, Linux terminology modules, and being able to load and unload them at runtime whenever they are needed adds another dimension of flexibility to Linux.

1.3.3 Linux kernel versions

Beginning with Version 0.1, a number of Linux versions have been released into the world and a numbering scheme was adopted. It originally started with a three number scheme and was later extended to a four number scheme.

- The first number denotes the kernel version. It was changed in 1994 and 1996 for version 1 and 2 respectively.
- The second number stands for the major version. Even numbers such as 2.4 or 2.6 are assigned to so-called stable releases that are suggested for use in production systems. As of this writing, the vast majority of productive Linux systems are based on the 2.4 and 2.6 kernel versions.
- The third number serves as a minor version. In the past, during the original three number scheme, this number was increased whenever a relevant patch, for example a security patch, was released. With the newer four number scheme, the minor number is only increased when new functions are introduced into the kernel.
- Finally, the last number is now increased with the release of system pertinent patches, whether they are bug fixes or security enhancements.

As mentioned, the vast majority of productive Linux systems, no matter the platform, are now based on kernel version 2.4 or 2.6. The kernel version 2.4 introduced a number of enterprise relevant features, such as logical volume manager support and increased the filesize limit to 2 GB per file. Also, Java™ support was introduced in the kernel.

The kernel version 2.6 introduced major enhancements on both the workstation and on the enterprise level. Also, the efforts for building Linux for embedded systems have been merged with the mainstream work. The internal threading model of Linux has been changed and now supports the Native POSIX Thread Library, which is important for mission critical applications. Last but not least, the driver and module concept of Linux was hardened and optimized towards stability. The following is an summary of the more important enhancements of the 2.6 kernel:

- PCI hot plug now allows you to insert or exchange a PCI hot plug adapter into an PCI slot under a running OS. The adapter might be of the same or of an different type. The kernel assigns the new resources to support the adapter.
- The 2.6 kernel features a new algorithm to handle virtual memory, called reverse mapping. It improves the virtual memory handling under load.
- Symmetric Multi Processing (SMP): The scheduler of the 2.6 kernel addresses certain problems in conjunction with SMP. It also shows improved load balancing and improved performance of interactive applications.

- Kernel preemption: The 2.4 kernel does not allow kernel processes being preempted. This also affects user processes executing system calls. The 2.6 kernel now allows system processes to be preempted.
- Threading model: As mentioned, the 2.6 kernel now implements the Native POSIX Thread Library (NPTL), which implements a 1:1 thread model, where one kernel thread is assigned to one user thread. This will greatly improve Web service and Java applications. The number of process IDs has been increased from 32000 to over a billion task IDs.

1.4 Linux and POWER

This section focuses on the POWER architecture. We will discuss Linux distributions that are used on the POWER architecture and discuss some of the features POWER systems offer.

Linux distributions for the POWER architecture

The Linux kernel itself, along with a number of tools and applications, is freely available under the GNU General Public License (GPL), which can views at the following Web page:

http://www.gnu.org/copyleft/gpl.html

Anyone can access the kernel source and compile a Linux kernel. However, the kernel by itself is of little use, as it is missing a number of key components, such as useful device drivers, application and development interfaces, and so on.

While the GPL does not allow somebody to charge for Linux, there may be a charge for additional efforts for making a full operating system based on the kernel, such as providing printed documentation, a set of device drivers, an installation and deployment routine, and support. The term “distribution” is now widely accepted as the term for a working package of Linux. The source code to a given distribution is available, and is usually included with the distribution media. A distribution is compiled for a given platform, such as Intel x86 or POWER. Because a given distribution is normally available for multiple platforms, all distributions are generally source compatible.

Currently, more than 100 profit and non-profit organizations provide Linux distributions for a number of different platforms. IBM is working with Red Hat, Novell SUSE, and ASIANUX to provide Linux for IBM platforms, including those platforms built on the POWER architecture. IBM itself does not provide a Linux distribution, but customers may order their Novell SUSE or Red Hat distribution through IBM.

Examples of commercial distributions

Red Hat, Inc. (http://www.redhat.com) is among the best known Linux distributors. In 2003, Red Hat Enterprise Linux (RHEL) AS3 for POWER became available for IBM eServer™ pSeries®. In 2004, RHEL AS3 was updated to support the IBM System p5™ server, OpenPower servers, and JS20 blades. This distribution is based on a Linux 2.4 kernel with a few features ported from a Linux 2.6 kernel.

The current release of RHEL AS4 is based on the 2.6 kernel and supports POWER4™, POWER5, and POWER5+ based servers, and JS20/21 blades. More information can be found at the following Web site:

http://www.redhat.com/software/rhel

Novell SUSE provided the first Linux distribution for the pSeries and was the first distribution for RS/6000® servers. The latest Novell SUSE Linux distribution for enterprise clients is called SUSE Linux Enterprise Server Version 10 for POWER and was released in August 2006. It is based on kernel version 2.6 and supports 32-bit and 64-bit applications. Version 8 of the distribution, SLES 8, is still available and supports POWER4 servers and JS20 blades. For more information, please refer to the following Web sites:

http://www.novell.com/products/server/overview.html

Community supported distributions

Besides the IBM supported enterprise distributions, a number of community supported distributions for the POWER architecture exist, such as openSUSE, Fedora Core, and Debian. For more information about these distributions, please refer to the IBM Linux on POWER wiki at the following Web site:

http://www.ibm.com/collaboration/wiki/display/LinuxP/Home

1.4.1 Running Linux on POWER servers

As mentioned, Linux runs on servers based on POWER processors. Also, starting with the POWER4 family of servers, support for running Linux in logical partitions (LPAR) was introduced. Logical partitioning allows the creation of a number of virtual Linux systems running on a single server. Every logical partition provides the required resources, such as disk space, memory, a network connection, and a CPU for the Linux installation running in that partition. Figure 1-3 on page 13 shows an example of a typical Web environment consisting of two Web servers, a back-end database, and a test environment. Normally these resources would run on four separate servers that provide the required hardware for the applications.

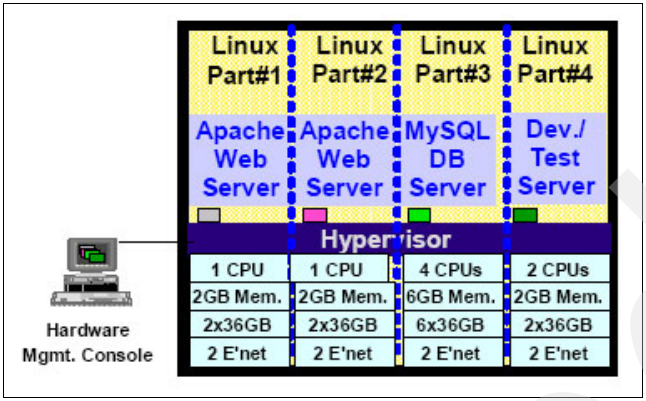


Figure 1-3 Example of Linux running in logical partitions

With logical partitions, the whole environment can be consolidated into one single piece of hardware. On POWER4 servers and Linux systems based on a 2.4 kernel, only static LPARs are supported. Static means that after any reconfiguration occurs, the Linux OS running in an LPAR must be stopped and restarted after reconfiguration.

Starting on POWER5 and POWER5+ systems, dynamic LPAR is supported with Linux systems running kernel 2.6. Dynamic LPAR is not supported with the 2.4 kernel. The dynamic LPAR feature allows the reconfiguration of logical partitions without stopping the partition.

Another feature introduced with the POWER5 systems is the introduction of I/O virtualization through a Virtual IO server. It allows systems running in multiple LPARs to share system resources, such as a single Ethernet adapter or a single SCSI adapter. In addition, it allows the creation of VLAN inside the server, thus allowing communication among LPARs without a dedicated network card.

Figure 1-4 on page 14 shows a server using a VIO partition.

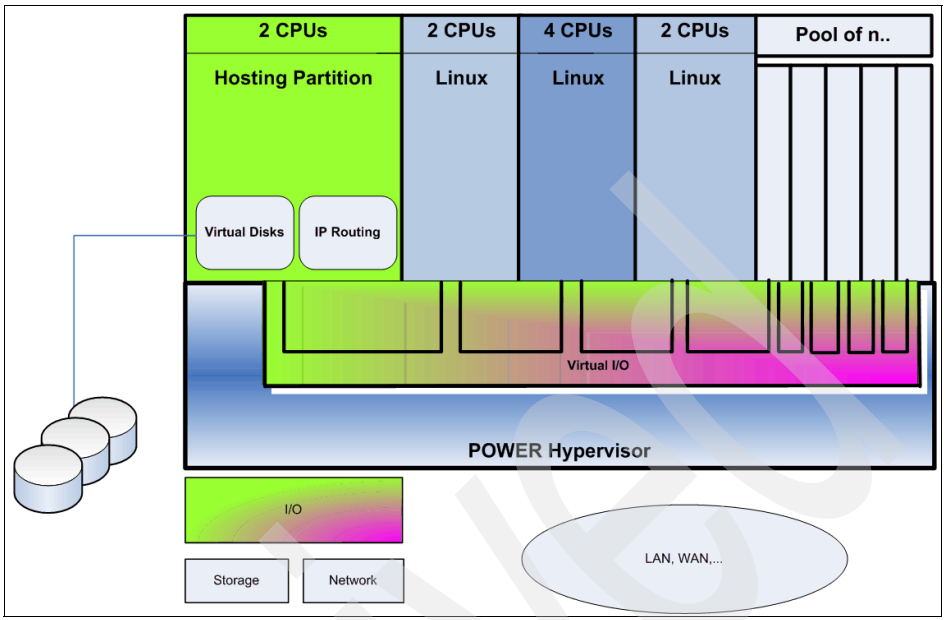


Figure 1-4 Example of a server utilizing a VIO partition

The virtual IO server owns and controls all the I/O resources and assigns virtual I/O devices, such as disk storage through an virtual SCSI adapter and network connectivity through an virtual Ethernet card. This allows, for example, sharing a single physical network adapter among all the virtual machines. Also, the communication between virtual machines running in an LPAR without a dedicated network adapter is supported.

1.5 Linux and the enterprise

The initial versions of Linux were developed for people that wanted to run an UNIX-like OS on CISC boxes without any concern for high performance, high availability, and other business needs. A few years later, some companies upgraded Linux to a complete business solution that was intended to compete against UNIX and Windows® on the market.

With over 10 years of development and improvements, some Linux versions, such as Red Hat and Novell SUSE, have already proved that their distributions can be trusted for use in business and can be an less expensive option. They opened an new market share to companies that want reliability and security but want to expend as little capital as possible for an operational system.

Some companies are already using Linux versions of the most popular application and database software used by businesses today, giving security to customers that want to use Linux as their operating system. Because of these developments, many customers are slowly migrating to Linux, forcing other companies to deliver software that is compatible with Linux as well.

A good example of this growing shift to Linux is the impact on the market share of Web servers. The number of Linux servers that are running Apache is growing rapidly, almost every month. Figure 1-5 shows the Web server market share.

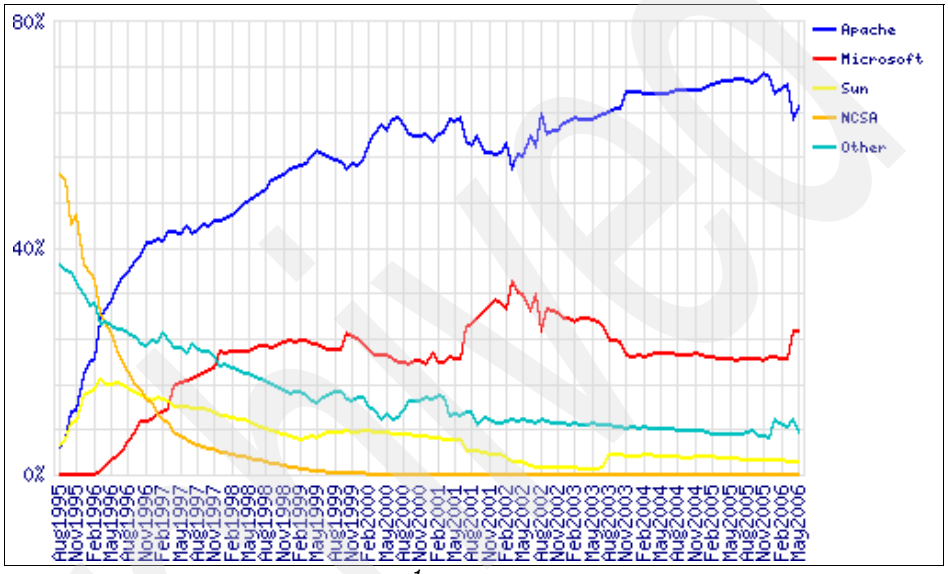


Figure 1-5 Web Servers Marketshare[1]

Another concern of customers is support for the Linux OS. Although support was problematic in the past, many Linux companies today give good support to their customers, and there are now professional certifications for Linux systems.

Given all these points, Linux is certainly suitable and desirable for enterprise level systems.

[1] http://news.netcraft.com/archives/2006/05/index.html

2

Solution requirements

In this chapter, we describe the requirements for a successful deployment of applications under Linux on POWER. These requirements are not unique to a Linux or a IBM System p environment. As you plan your system, you need to look at software options, storage options, and to what level are you going to perform system management tasks.

2.1 Requirements

This section discusses the requirements to deploy a High Availability (HA) Solution on Linux with POWER, using examples for Small and Medium Business (SMB) and Enterprise (Large Business) customers.

2.1.1 Hardware requirements

The hardware requirements for a HA database using DB2® depends how you planned your database solution. Solutions with one or two databases and information about clustering or file replication can be found in 3.4, “Database planning” on page 46.

Storage

Using one or two databases, you will need at least one server to store your databases. IBM has a complete line of storage products for SMB and for Enterprise Business that work with Linux.

The Enterprise Business solutions include the Enterprise Storage Server® (ESS) Family and the IBM TotalStorage® DS Family, which was formerly called the FAStT Storage Servers Family. Some highlights of both storage families are:

- Designed to simplify storage infrastructures, support business continuity, and improve information life cycle management
- Focused on providing significant improvements in functionality, performance, and total cost of ownership by leveraging leading IBM technology
- Offers a range of scalable solutions to address your storage needs, from the smallest distributed storage server to the largest data center
- Provides a continuum of enterprise storage products that support IBM as well as non-IBM open systems and mainframe servers
- Offers excellent price/performance options with a broad array of products that serve as a foundation for tiered storage environments
- Enables streamlined systems management with common management tools based on open standards
- Offers a broad family of disk storage solutions that provides the freedom and choices to best match the value of data with the most appropriate storage

A good example of storage devices for SMB that is compatible with Linux on POWER is the DS4300, formerly FAStT600 Storage Server (1722-60U/6LU). This device uses Fibre Array Storage Technology to improve performance and capacity to 2 Gbps Fibre Channel and 2 TB of physical storage capacity. This

storage device supports RAID 0, 1, 3, 5, or 10, and can be used with other OSes, such as Microsoft® Windows, AIX®, HP-UX, and Solaris™.

For Enterprise customers, you can use the DS6800, which supports 38.4 TB of capacity using RAID 5 or 10, and is compatible with Linux on POWER. The differences between the DS4300 and the DS6800, besides the capacity, are the copy services and availability features. You can see the differences in Table 2-1.

Table 2-1 DS4300, DS6800 comparison

Item	DS4300	DS6800
Copy Services	FlashCopy®, VolumeCopy, and Enhanced Remote Mirroring	FlashCopy, Metro Mirror, Global Mirror, and Global Copy
Availability Features	Fault-tolerant, RAID, redundant power/cooling, hot-swap drives, single/dual controllers, concurrent microcode, update capacity, and dual-pathing drivers	Fault-tolerant, dual redundant and hot swap RAID Controller Cards, Battery Backups Units, Fibre Channel switch controllers and power supplies, and nondisruptive hardware and software code loaded updates and multi-pathing device drivers

For other storage devices, go to the following Web site:

http://www.ibm.com/servers/storage/

Storage area network (SAN)

To connect the servers to the storage device, you will need an Storage Area Network switch, which will be the physical bridge between the servers and the storage device. This component must be high availability too, because when the application loses connection to a storage device that goes offline, the application or database becomes unavailable.

The first thing that you need to consider, besides the compatibility between the storage device and the SAN, is the number of applications that will use the storage device. With this information, and knowing that the applications are mission critical, we need to use two paths for each critical server to the storage device, and all the connections need to be calculated in order to determine the number of ports on the SAN that you will need.

We also need to consider high availability for the SANs in order to avoid a single point of failure on this device. For example, if you need 11 ports on the SAN to

connect all your Systems, instead of using one SAN with 12 ports, you can use two SANs with eight ports each, eliminating the single point of failure on the SANs and making it possible to load balance them, as shown in Figure 2-1.

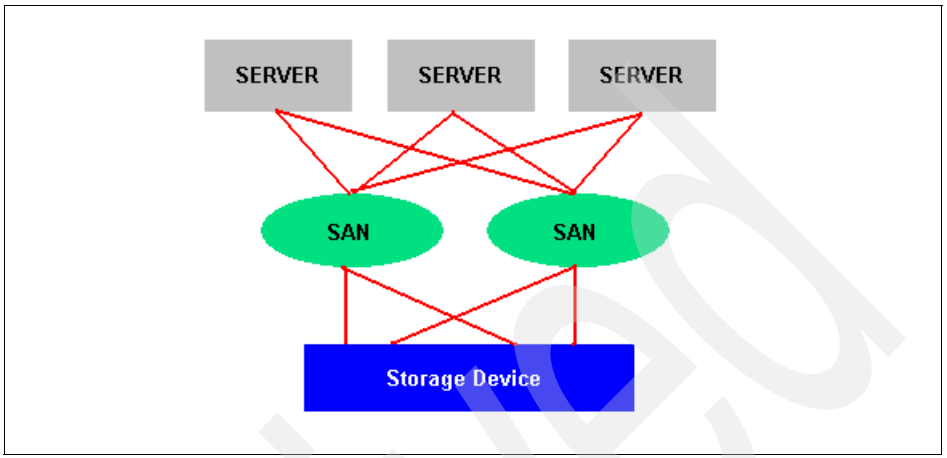


Figure 2-1 Using high availability on SANs

In this example, we can use two IBM TotalStorage SAN16-Bs, which have 4, 2, and 1 Gbps Fibre Channel and support Linux on POWER and other OSes. But if you need more than 24 ports, and more performance, scalability, and high availability, you can use an IBM TotalStorage SAN256B director, which has high availability with built-in redundancy that is designed to avoid a single point of failure. You can use 16 to 256 ports using eight blades, two control processors for HA and dual power supplies and blowers, in addition to the hot-swap components.

For more information about the IBM TotalStorage SAN and CISCO MDS series, go to the following Web site:

`http://www.ibm.com/servers/storage/san/`

Network

To link all servers, storage servers, database servers, and other devices together, you will need an network component that can handle the data transfer between the servers and the communication between the clients and server. To understand the different networks of the solution, refer to 3.5, “Network planning” on page 51.

On the three networks, you will need the network devices (routers, switches, HUBs, and firewalls), cables, and NICs. These networks must be high availability or the solution will not be an complete HA solution. A good strategy is to use

FDDI between the servers (server network), and Ethernet with redundant switches (access network).

The FDDI technology can remove the single point of failure in the network, but cannot remove the single point of failure of the NICs, so we recommend using two NICs and each server, as well as a Linux script to migrate the configuration to the backup NIC.

There are many devices that can be used to create an network environment, but be careful to choose the right devices that fit your utilization requirements, and remember that these devices must support high availability.

If the desired solution includes some services to be provided through the internet, you should use at least two firewalls to create an demilitarized zone (DMZ) between the service provider machine and the final database or application. This firewalls can be two Linux machines running on POWER machines. Figure 2-2 shows an example of a DMZ.

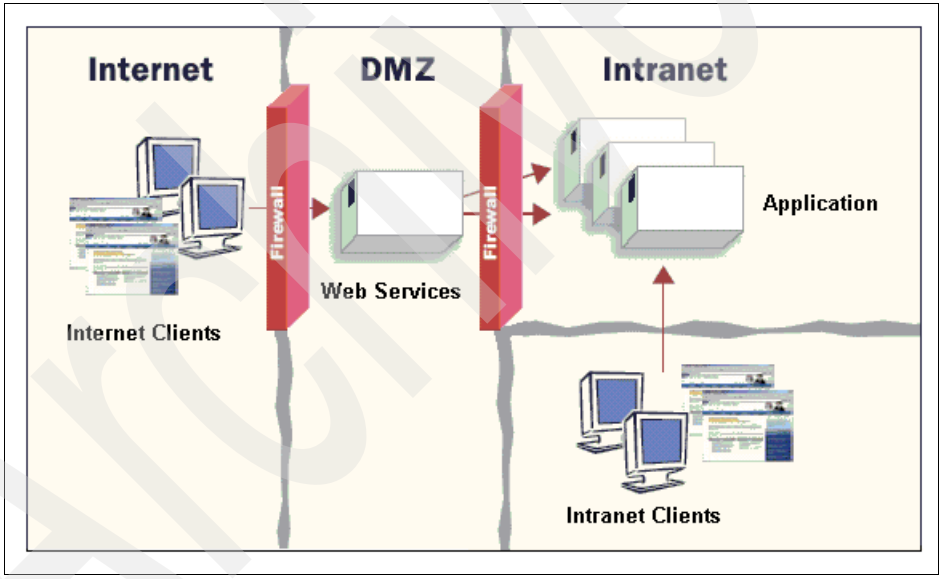


Figure 2-2 Example of DMZ network

If you have plan to have a Virtual Private Network (VPN) server to accept connections from outside the internet network, we recommend that you have a VPN server inside the DMZ. Remember to use the “new” users to measure the network size and throughput.

Also, regarding the access network and the Ethernet redundant switches, be sure to load balance between the applications servers, thereby increasing the performance and avoiding connection problems if the applications fail.

Server

The types and number of servers depends on the desired solution. We show the capacities and features of the POWER boxes running the Linux OS. To learn more about which servers and how many of them are needed for your solution, refer to Chapter 3, “Solution methodology” on page 29.

Keep in mind that the POWER servers can use Logical Partitions (LPARs), which means that you can have one server (box) that contains more that one system using the same hardware. So if you plan to use LPARs, you can avoid a single point of failure for your solution by having the two application servers of your high availability solution on different boxes, as shown in Figure 2-3.

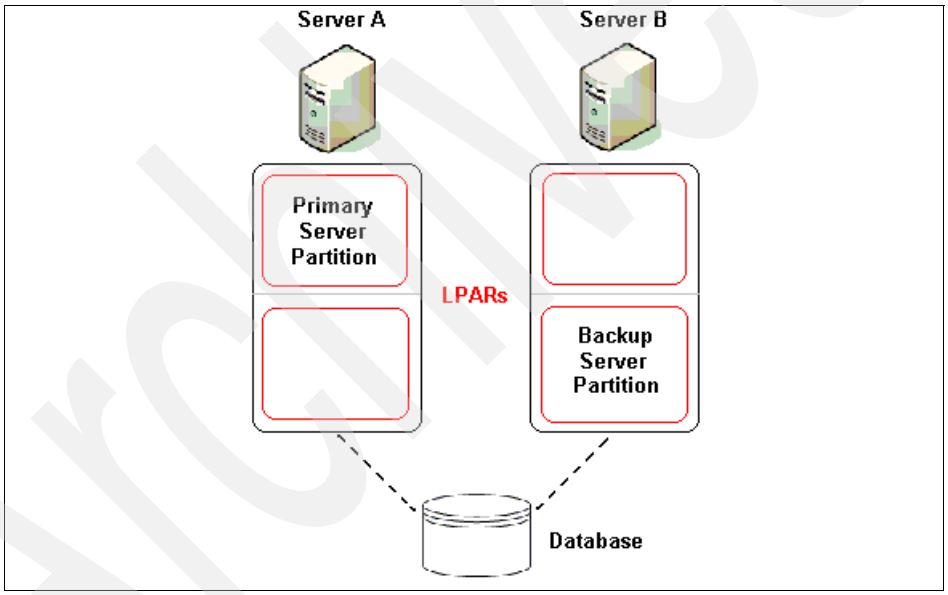


Figure 2-3 Using LPARs to HA Solutions

Should you use IBM System p5 servers, you will be able to use features like the following:

- Reduce complexity through centralized management and administration and pooling of resources
- Improve utilization rates by sharing a pool of virtual resources and micro-partitioning

- Automate processor balancing to address business peaks
- Integrate and access business processes with secure inter-partition communications using virtual LAN
- Extend i5/OS® storage automation and management to multiple operating environments using Virtual I/O
- Run multiple independent operating environments and business workloads, including multiple languages and time zones for global IT infrastructure consolidation
- Consistent expansion: Add what is needed when it is needed

For the SMB (p5 systems for entry and mid-range customers) we can use, for example, two IBM System p5 550Q Express servers that have four or eight IBM POWER5+ Processors with 1.5 GHz.This server can use one to 64 GB of system memory and up to 31.2 TB of Internal Storage.

In a benchmark example of the System p5 550Q Express server, if it is used to support an SAP ECC 5.0 environment, it can handle an database of 220,000 MB with 1,000 users, running Linux (SLES 9) with DB2. It can process 301,000 dialog steps per hour or 100,330 fully processed line items per hour, and the average response time per dialog step will be below the SAP recommendation (below 2 seconds each). This values and times are published in Section 8, “SAP Standard Application Benchmark Published Results” of the IBM System p5, IBM eServer p5, pSeries, OpenPower, IBM RS/6000, BladeCenter Performance Reports, March 21, 2006.

For larger databases and systems, you can use the IBM System p5 595 server, which uses the fifth-generation 64-bit IBM POWER5 processors that can be configured up to 64-way symmetric multiprocessing configurations (SMP). This server can deliver exceptional performance, reliability, scalability, and flexibility for a full range of complex, mission-critical applications with demanding processing requirements, including:

- Business Intelligence (BI)
- Enterprise Resource Planning (ERP)
- High Performance Computing (HPC)

You can also use all features provided by the POWER technology combined with the Linux OS, such as:

- POWER5 microprocessor
- High memory and I/O bandwidth
- Flexibility in packaging
- Shared processor pool*

- Micro-Partitioning*
- Virtual I/O*
- Virtual LAN*
- Dynamic logical partitioning*
- Mainframe-inspired RAS
- Broad range of CoD offerings*
- Grid Computing support*
- Scale-out with CSM support*
- High Performance Switch attachment*
- Multiple operating system support

* Indicates that this feature is optional, is available on selected models, or requires separate software.

To understand the possible utilizations on HA solutions, please refer to 4.2, “HA scenarios and examples” on page 58 or go to the following Web site:

http://www.ibm.com/systems/p/

Backup

Your solution must include one tape library to manage and execute the backup and restore procedures, including off-site backups. With these backups, you can restore your database in case a logical deletion occurs (for example, a drop table), or even restore your database on another site, or do site backup tests.

This library must support the Linux OS running on POWER server, for example for an SMB, you can use the TS3310 Tape Storage. This device can have up to six LTO drives for read/write on tapes and accept 122 cartridges (102.4 TB). You can also use the 2 Gb Fibre Channel or SCSI-LVD with the Linux OS.

This tape library can be bought in small parts, using the IBM scalability feature to grow the storage capacity together with your demand of new storage tapes. This kind of solution can decrease the TCO of the solutions.

The tape library does not have to be highly available, but if you plan to create an site backup or prepare for Disaster Recovery with the other tape library, make sure that the tapes can be read on both systems. The tape compatibility list can be found at the following Web site:

http://www.ibm.com/servers/storage/media/index.html

To enterprise customers, you can use the TS3500 Tape Library for an normal tape storage operation or use the IBM Virtualization Engine™ TS7510, which is

the first member of the IBM Virtualization Engine TS7000 Series. The TS7510 combines hardware and software for virtualization of tapes to open systems servers connecting over Fibre Channel connections. Using disk technology and tape technology to virtualize or emulate tape libraries, media, and drives decreases the TCO of the solution too.

For more information about the IBM tape libraries, go to the following Web site:

http://www.ibm.com/servers/storage/tape/

Servers to monitor and administrate

We will not use examples of servers for this topic, because the monitor server and the administration server have many differences in their applications' environment, which means that each application (SAP, PeopleSoft, Siebel®, and so on) has different ways to be monitored and administered.

Instead, we want to remind you that the solutions will need to be monitored, and, in order to avoid impacting the business applications, separate server should be used for hosting the monitoring software. We also suggest that the administration server be put on a separate server from the applications (and it could be on the same server as the monitoring software) and inserted into the server network.

Imagine if it was necessary to execute an online reorganization of a big table/tablespace (300 GB) that is connected to the intranet of your company through an normal workstation. This process could be execute for hours, using all the available network resources, and not be fast enough due the workstation's capacity. Instead, a server could be used for this task and other administration and monitoring actions, improving the execution time.

2.1.2 Software

Software is a key component to a highly available, high performance solution.

Operating system

To create an high availability environment, we need at least two servers (boxes) to avoid single points of failure in the hardware, which means that you will need the same number of licenses for the OS. You also need licenses for the monitoring and administration servers.

IBM hardware has been developed to work at least with two Linux versions, Red Hat Enterprise Linux and SUSE LINUX Enterprise Server, so you a have choice of which one to use. Keep in mind that each version has its own features that could be different from each other. For more information about these two distributions, please refer to the following Web pages:

- RHEL

  http://www.redhat.com/rhel/

- SLES

  http://www.novell.com/products/linuxenterpriseserver/

Database software

Your database software should be chosen carefully. There are more then 10 different RDBMs that exist, but not all of them have high availability features, so be sure that the database that you choose has those features.

Attention: Some ERP software vendors, like SAP, already ship the database software (RDBMS) inside the SAP solution box, including this software, which is included in the final cost.

Popular database software includes IBM DB2 Universal Database™ and the Oracle® Database. We recommend these products for your highly available solution. To learn more about these products, please refer to the following Web sites:

- Oracle

  http://www.oracle.com/index.html

- DB2

  http://www.ibm.com/software/data/db2/udb/edition-ese.html

Monitoring and administrative software

All this hardware and software needs to be administered and monitored, which means you will need software to perform these functions. Many products can be used to monitor and administer your environment, but IBM recommends IBM Tivoli products. The Tivoli family of software was developed to work with almost any database, operating system, and ERP system. All your software can be monitored from one single tool that sets the same thresholds and rules for the whole environment, thus easing your monitoring and administrative demands.

To read more about the Tivoli family of products, please refer to the following Web site:

http://www.ibm.com/software/tivoli/

Other software

Some other software could be necessary to create the high availability environment besides the storage offering, RDBMs, and the application itself. For example, some heartbeat software might be used between the primary and

backup server to advise you of areas of concern. You can read more about this additional software in 4.4, “HA solutions” on page 73.

The top vendors of BI, ERP, and HPC software already include some features to use HA in their software, so ask your vendor about these features and how they can be used in your solution.

3

Solution methodology

This chapter will provide information about how to plan a mission critical installation of an SAP Netweaver system, and will show all the information about the SAP variants that can be used, and how to distribute them across the servers.

3.1 Solution prerequisites

Here we discuss the different approaches to sizing an SAP installation using examples of data to understand the SAP sizing workflow. SAP provides a tool to help the customer with the first sizing of the environment, and this process is called the Initial Sizing, which is used in the first step of the project to discover the initial size of the required servers. After the installation and all developments, authorizations, customizing, and configuration are done, SAP recommends other sizing, such as Expert Sizing, which is executed on the running environment to measure the utilization and to discover if this environment needs to be resized.

More information about SAP sizing can be found at the following Web site:

http://service.sap.com/sizing

In the second part of this chapter, we will talk about the ERP solutions based on HTTP connections, like Siebel and Peoplesoft. Both software applications and their originating companies are now part of Oracle.

3.1.1 Customer data

To execute the SAP sizing tool, we need to map all the business requirements of the customer. This step is very important, because the information is used by SAP to calculate the size of the servers. Wrong or incomplete information can impact the performance and scalability of the servers.

The most important required information for the SAP sizing tool is:

- SAP software version
- Database version
- Operating system version
- Hardware vendor
- Number of users
- Amount of reporting (dialog and background)
- Load profile
- Customizing

The four first entries will be used for the calculation process, and can be answered quickly, but the other s must to be studied and measured correctly. The number of users or user definition are the number of users that will use the system at the same time, not the Named Users (users that have an login on the system), so be careful to not use the wrong number.

SAP also has three categories about the typical activity patterns of users, assuming a working week of 40 hours. The categories are:

Low User Users that only use the SAP system a few times a day (10 dialogs steps an hour, or 1 in 6 minutes).

Medium User Users that use the SAP system for a few hours every day (120 dialogs steps an hour or 1 every 30 seconds).

High or Power User Users that use the SAP system the entire day for an 8 hour work day (360 dialog steps an hour or 1 every 10 seconds).

The amount of reporting items is the number of pages that will be requested by the SAP systems each day, which includes all the spooling generated by the system. This data is very important for measuring processor sizing, because the background processes use more processor threads when they are running without user intervention.

For the Load profile, it is necessary to know the peak hours and determine the hardware specification as well, because in addition to the actual utilization of resources, the SAP sizing tool will need to increase the final size to avoid problems with growth.

Customizing is the number of modifications of the standard transactions that will be utilized by the customer.

All this information and numbers will be translated into technical requirements by the SAP sizing tool, which will result in a hardware independent server size number called SAPS. SAPS means SAP Application Benchmark Performance Standard.

For example, 2,000 fully processed order line items per hour need 100 SAPS to be processed, which represents approximately 6,000 dialog steps or 2,400 SAP transactions. To calculate the SAPS, we can use two sizing approaches, based on user numbers or based on throughput:

User numbers This approach is quite easy to use, and works well only for small systems, because it is too risky to plan the capacity of a large system with just a user-based sizing model.

Throughput This approach requires more effort than the user-based model, but is better able to determinate CPU and disk utilization. It is usually used for large systems and recommended by SAP.

3.1.2 Executing the SAP sizing tool

After choosing our sizing approach and finishing the mapping of all the business requirements, we need to run the SAP sizing tool on the SAP Web site at:

http://service.sap.com/quicksizing

Click the item **Start Quick Sizer** in the left side menu, and a Web page will load, as shown in Figure 3-1.

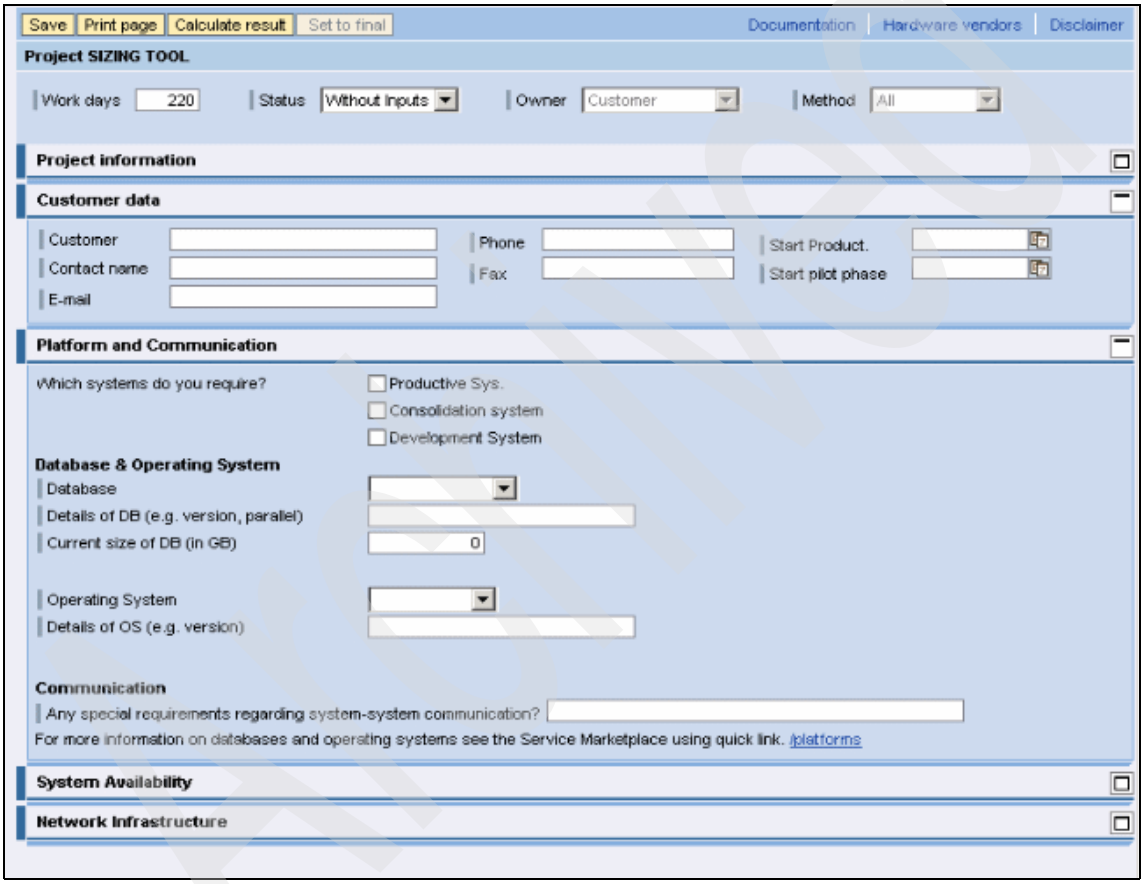


Figure 3-1 SAP Sizing Tool Questionnaire

3.1.3 The results

Attention: Fill out all the form carefully with accurate and up-to-date data and make sure that you and the customer use the same user definition, because the results can only be as good as the provided information.

Choosing **Calculate Result** once you have filled in all the information required by the questionnaire will give results for the entire project on different levels and with many available views that go from an overall result to the statistics for the 30 largest tables. Be careful with decimal displays, because they depend on your language settings.

The results will be displayed for both the SAP and Disk values categories, and are shown in Table 3-1. If your CPU size exceeds 20,000 SAPS or the disk utilization exceeds 400 GB, the tool will ask you to contact the SAP.

Table 3-1 SAP sizing result categories

Category	SAPS	MB
XS	1,000	70,000
S	4,000	160,000
M	8,000	220,000
L	14,000	320,000
XL	20,000	400,000
XXL	Contact SAP or hardware vendor.	

With these results, we can go to our hardware vendor to translate the SAPS to MIPs and find the exactly machine to fit on our needs. You can use the tools on the SAP Web site at http://www-912.ibm.com/wle/EstimatorServlet, which helps to size machines based on SAPS estimates.

3.2 Planning the SAP landscape

In this section, we discuss the needs of a mission critical SAP environment so we can understand the methodology of the SAP transport system across the entire landscape, and explain each SAP component so we can understand how they work together.

3.2.1 Defining the systems

The SAP system can be used in modules. It creates a specific module for each common specification area, such as Management Materials (MM) or Sales Distribution (SD), but all these modules need to be adapted or customized to work the way that the customer needs. This customization usually is an alteration of the saturator ABAP™ code or a new code. The code alterations could be harmful or even dangerous to the system, so all this code changes cannot be done directly on the production systems. SAP recommends that you create a landscape that includes development, quality, and production systems and use the authorization transport tool to migrate these code changes across the systems.

The systems' hardware specifications between the systems are different because they are utilized differently. To understand these concepts, refer to Figure 3-2.

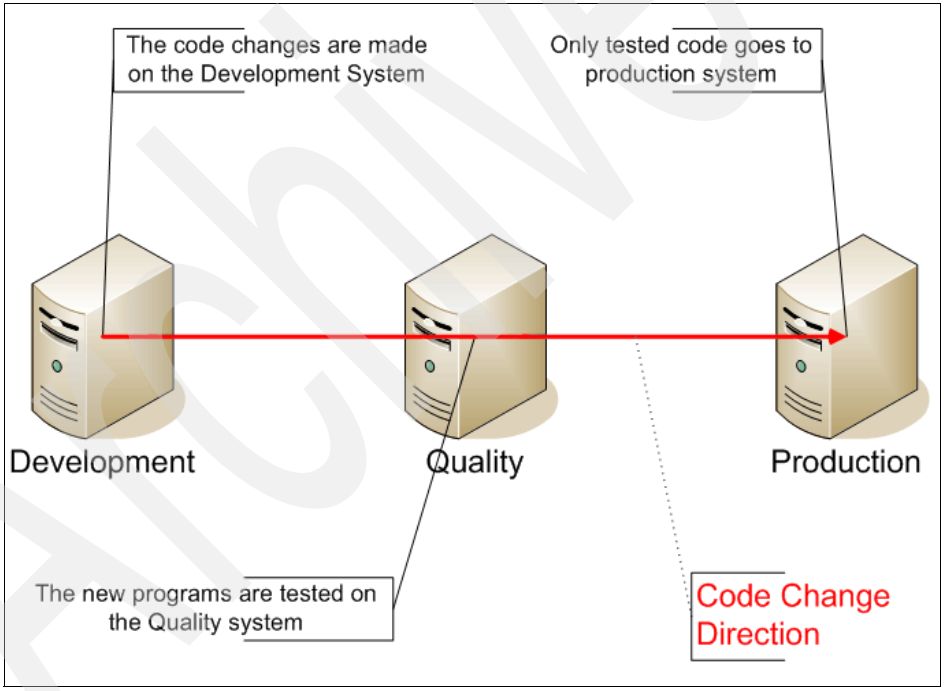


Figure 3-2 SAP systems landscape example

Using landscapes, we can have an single point for managing changes and modifications. Some customers choose to use only two systems: one for development and quality control, and the other for production, but this is not recommended by SAP.

The development system have two different phases. At the beginning of the project, after the initial installation, the consultants need to use the systems for customizing, configuring, and testing of features for a long period of time. After this phase, the utilization of the development system will be limited to corrections of the programs or, depending on the modules of this environment, some configuration. For example, if you have Human Resources (HR) and Financial (FI) modules installed on your SAP environment, and if the financial laws of the local government change, then the FI module needs to be changed on SAP too. The final average utilization depends on the modules and new projects of this SAP landscape.

The size of the database of the development system are normally very small compared to the quality and production systems, because the consultants and developers do not need to have all the customer data to create or change programs.

The quality system must be as similar as possible to the production system. If possible, a copy from the production to the quality system should be executed frequently to guarantee that all new code can be executed on the production system. Therefore, this system needs to be on hardware that is able handle the database of the production environment and the tests of the developers and consultants. The main difference between the quality system and the production system it is that the quality system does not need to handle all the customers' work.

The production system must be powerful enough to guarantee the response times agreed to in the SLA and the hardware needs to avoid have a single point of failure. To accomplish this task, the SAP provides support to almost High Availability solution available.

Some customers also have training systems that are to provide training for new employees of the company. This environment is normally an small database, used for short periods of time for only a few concurrent users.

The Solution Manager

The Solution Manager in new SAP systems is usually called SOLMAN. In new SAP solutions, or mysap Solutions, based on Netweaver technologies, SOLMAN is a prerequisite. Without this system, you cannot generate the response installation key and will not be able to install it.

SAP is encouraging customers to use this new system to administrate the SAP environment. Besides generating keys, SOLMAN contains all the SAP features that was originally executed on Online Service System (OSS). The latest version, SOLMAN 4.0, has many new features:

- Issue Management and Tracking
- Bi-Directional interface to SAP
- Service Desk Application
- SOLMAN Diagnostic
- SAP Monitoring
- Roadmaps
- Digital Signature

Because of these new requirements, a new server will be required to install SOLMAN. Depending on how you use this system, a smaller hardware system can be used to support this application, but if you are planning to use all new features like monitoring and diagnostics, its possible that this system will be as important as the production system and will also need a HA solution.

3.2.2 Choosing the instances and their hosts

When we are installing only the ABAP system, we have three variants to use: Central, Distributed, and High Availability. The first two variants are usually used in the development and quality systems and the last one is used in the production system.

The *Central* installation only needs one server. If you install the ABAP Central Instance and the Database Instance using this variant, the SAP system will be not be protected against single point of failure.

Figure 3-3 on page 37 shows the Central ABAP system.

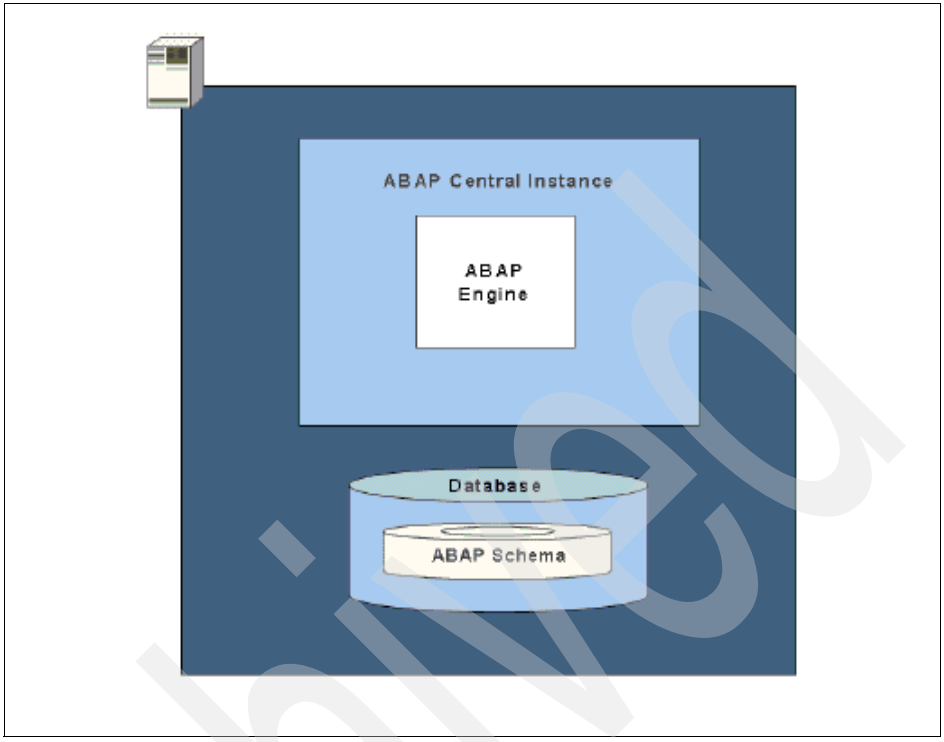


Figure 3-3 Central ABAP system

Attention: This installation is not recommended for production systems.

The *Distributed* variant was developed to allow the customer to install the central instance on an different server than the database instance. Using the distributed installation, we can use some High Availability features for the database instance.

SAP can be used with the most widely used RDBMS of the market, but the SAP software has the best compatibility with DB2. The transaction DB02 (SAP Database Maintenance) is completely different, because IBM and SAP worked together to make a complete solution available for DB2 administration inside the SAP system that is integrated with all the standard components of the SAP software.

Figure 3-4 shows the Distributed ABAP system.

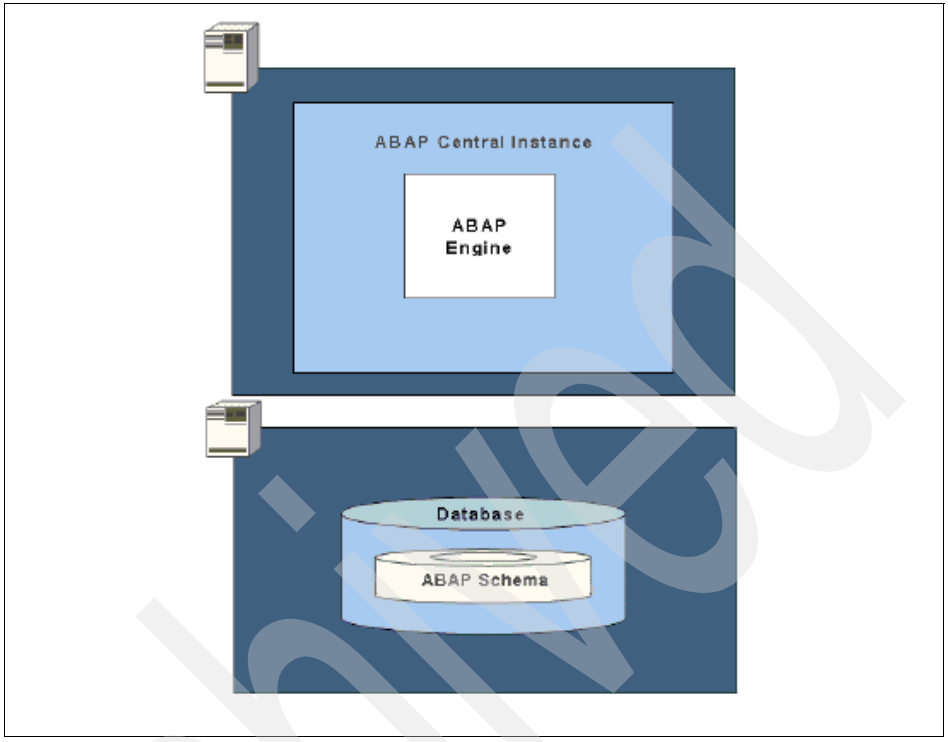


Figure 3-4 Distributed ABAP system

This variant also can be used to switch over the database instances on the backup server (ABAP Central Instance), thereby removing the single point of failure from the database server.

The recommended installation for production systems is the *High Availability ABAP System*. This installation was created to remove the single point of failure from the ABAP Central Instance by using the SAP Central Services (SCS) on an different server than the ABAP Engine.

Figure 3-5 on page 39 shows the HA ABAP system.

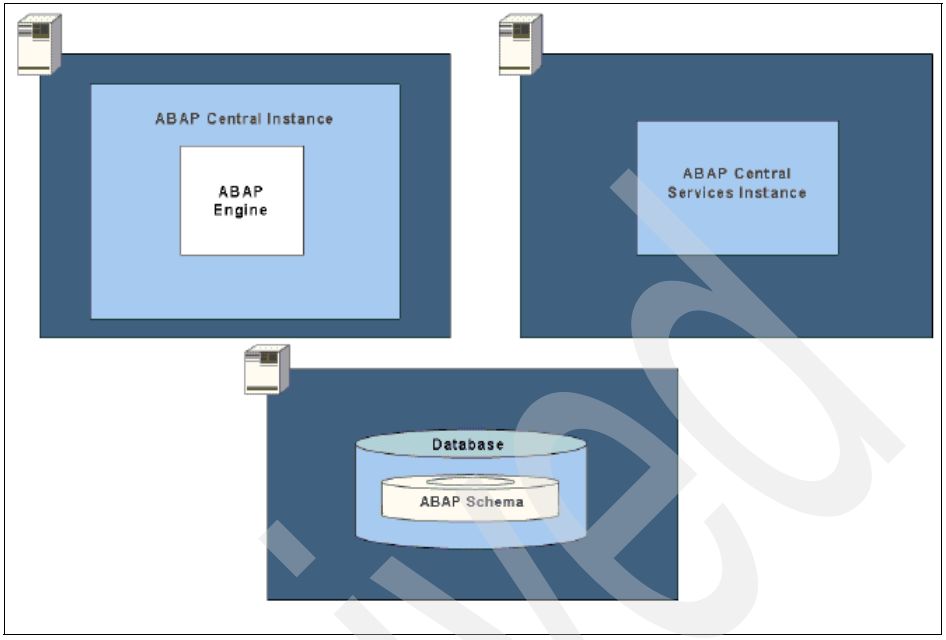


Figure 3-5 High Availability ABAP system

This way, both SAP instance nodes could switch over to the remaining node, removing the single point of failure of the Central instance installation.

Important: SAP recommends switchover clusters to archive High Availability SAP systems.

The same three options can be used on the Java installation, but the problem is that J2EE™ and their executables are not as ideally suited in terms of memory utilization as the ABAP installation is for this situation. Therefore, be careful with this installation and be sure that the sizing was made correctly.

At the beginning of the Java installation, at least 2 GB of real memory will be required; depending on the J2EE utilization, this number could grow quickly to 4 to 6 GB.

However, the most complex installation is the $ABAP + JAVA$ installation. This installation option is used to enable the security and reliability of the ABAP and the Web features of the J2EE.

Figure 3-6 shows the Central ABAP + Java system.

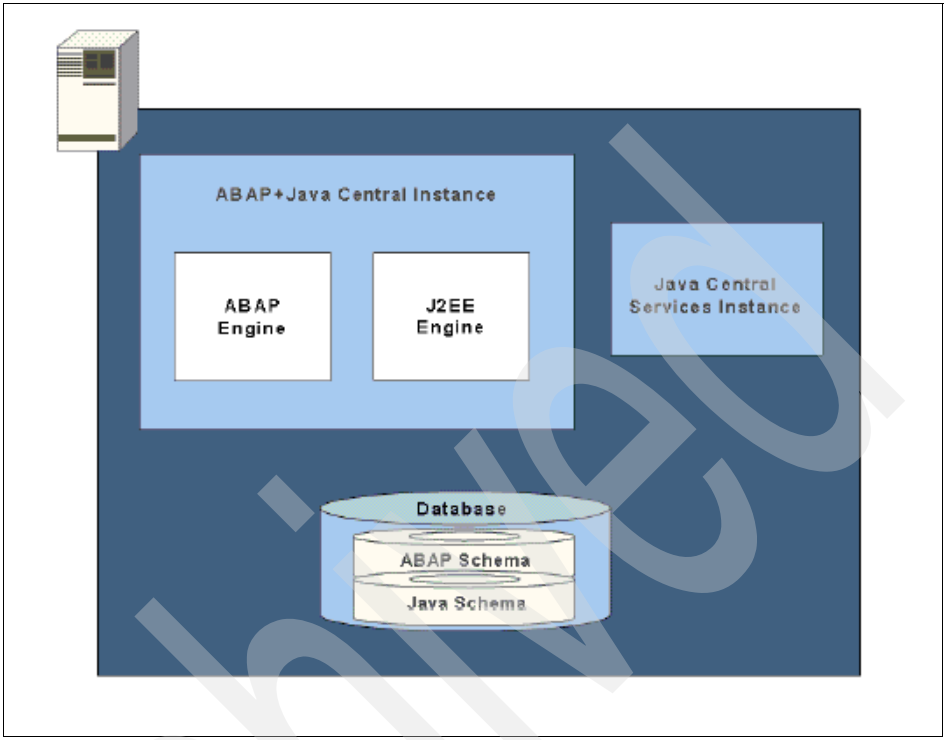


Figure 3-6 Central ABAP + Java system

To do the SAP installation using this variant, be prepared for high memory utilization and large I/O wait numbers. As we can see, the two database instances will be installed on the same binary database installation; even though the Java schema is smaller then the ABAP schema, this two databases together will increase the I/O utilization to use the file system at the same time. On the other hand, the ABAP work process will compete with the JAVA processes for the real memory available; if you do not have enough memory, the process will start to swap memory and use more I/O. Be sure that you have a good storage solution or you will likely have an bottleneck at the I/O utilization.

To reduce the possibility of I/O bottleneck, we could install both databases on separate systems, avoiding concurrent process for disk resources, as shown in Figure 3-6.

The best installation scenario includes the High Availability of ABAP and Java instances both, but in different servers, as shown in Figure 3-7 on page 41. The database is on a separate server as well, and could be installed with a cluster

feature, reducing the number of single points of failure from the software point of view.

In this solution, we need to pay attention to the network file systems (NFS) and the network between the servers. There must be a dedicated LAN between them, because they will exchange a great amount of information. If you use the corporate network segment, the performance of the network and the systems' performance will be degraded. To plan your network, refer to Chapter 2, "Solution requirements" on page 17.

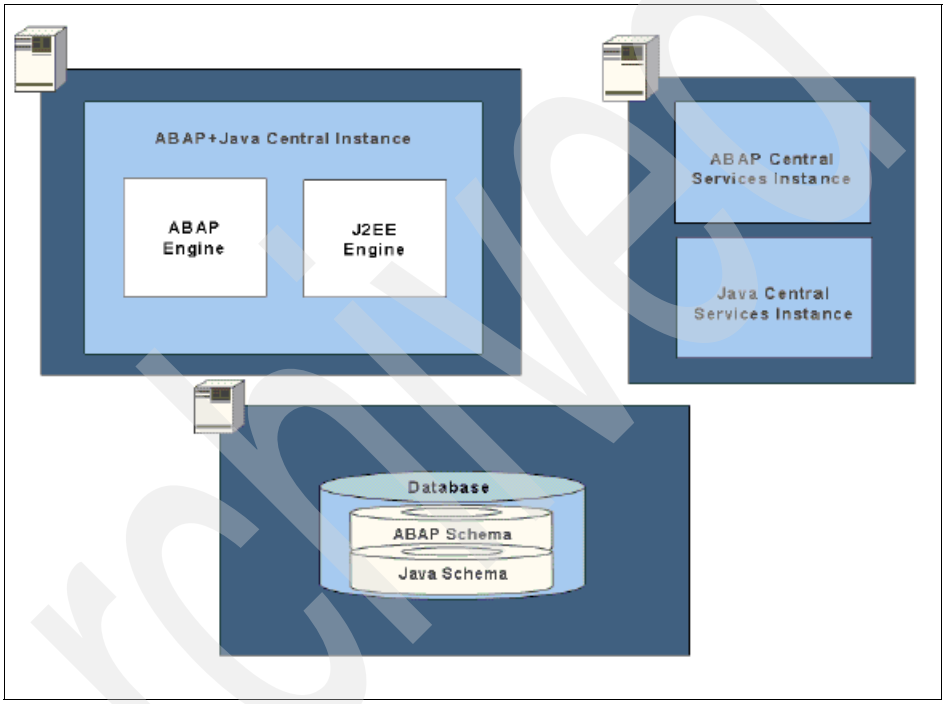


Figure 3-7 High Availability ABAP + JAVA System

The easy way to implement this solution is by creating multiple LPARs (Logically Partitioned Mode) in an IBM POWER server; just be sure to calculate the SAPS of each LPAR before installing the SAP systems.

3.2.3 Checking all prerequisites

Before starting the installation of your SAP system, we recommend that you run the *SAP Prerequisite Checker*, which will search for all the software and hardware prerequisites for the SAP installation. This feature is delivered with the SAPINST tool; to find it, go to the SAP system and select **Lifecycle Management** → **Preparation** → **Prerequisites Checker**.

3.3 Planning the HTTP based ERP landscape

In this section, we will talk about the application servers of the Siebel and Peoplesoft solutions. We will explain their features and how to use them on different servers to create an high availability environment.

3.3.1 Architecture

The first thing that we need to know is how each server application works and how can we split them into servers (boxes). Both solutions are very similar and use almost the same nomenclature for their applications. They are:

Web server The function of the Web server is to provide Web pages to the user using Web pages with Java servlet applications.

Application server The application server is the core of the architecture. It is responsible for all of the processing of the query results, as well as the administration and configuration of the ERP system for dialog utilization.

Database server The database server maintains the database and keeps it up and running, and is also responsible for all database SQL actions through the RDBMS software.

Process scheduler or report server
The process scheduler or report server are the background services provided by the ERP software and are responsible for generating all the reports from the background executions jobs.

3.3.2 Choosing the instances and their hosts

The installation architecture could be created based on the number of servers that you have to install for the solution. The Peoplesoft and Siebel applications could be installed on a single host on just one server. This kind of installation is only recommended for development and quality systems, because performance could be affected.

Figure 3-8 shows the single host installation.

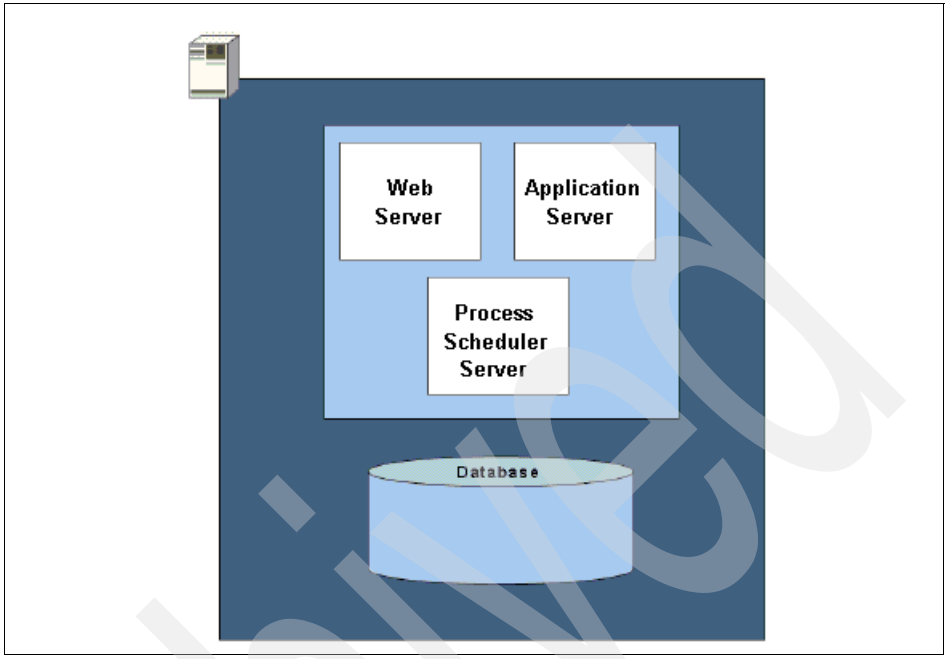


Figure 3-8 Single host installation

In some cases the process scheduler can be installed on one separate machine, usually because the server demands too much memory and processor time, or the customer wants the run the database on some UNIX platform and have the reports created by Crystal Reports or nVision software, that is, x86 based applications.

Another motive for installing the process scheduler or report services on separate machines is that the utilization behavior of the process scheduler is different from the other components. The process scheduler demands too much memory and processor time for small periods of time, usually when running financial reports at the end of the month. This utilization can affect the response time of the application to the users.

This component can also be installed on separate machines to create an high availability or load balance solution. The Peoplesoft and Siebel applications have features that allow you to use more than one of these components.

Figure 3-9 shows the process scheduler variant.

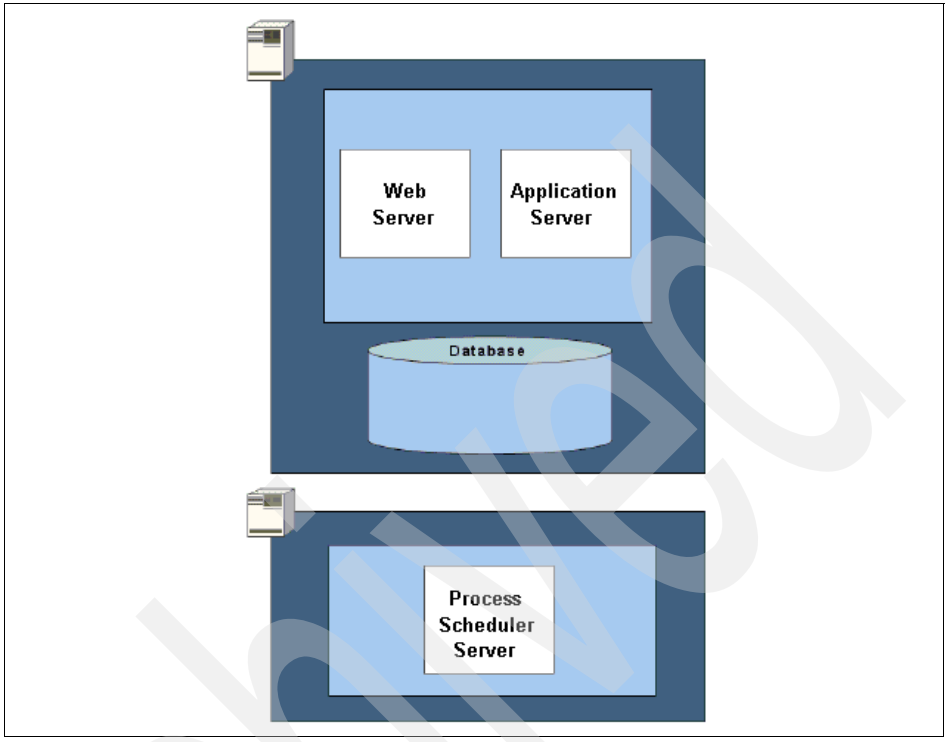


Figure 3-9 Process scheduler variant

Another common case for HTTP solutions allows access to your application through the intranet and the internet, so that business partners, customers, or even your employees can access the application when working outside the company.

The problem is that the Web server must be on the internet, but you should not have the database server there too. The best thing to do in this case is to separate the Web server from the other components and create an DMZ for this device.

You can also use two web servers for load balancing to get better performance, thereby increasing the response time of this component.

Figure 3-10 shows the Web servers load balancing variant.

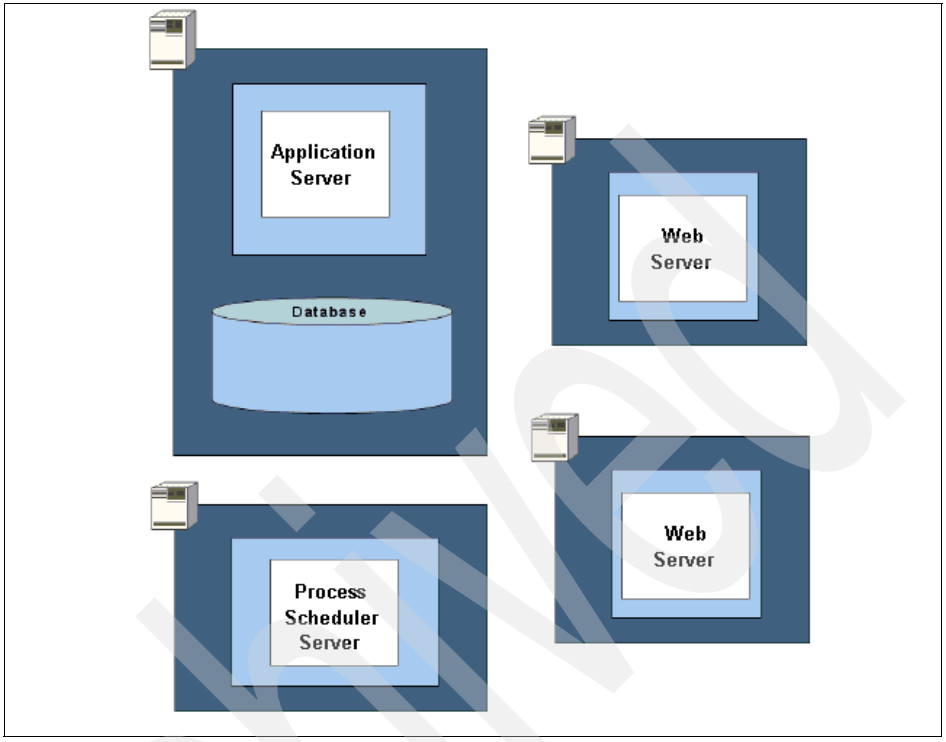


Figure 3-10 Web servers load balance variant

We can separate the database server from the application server, as with the other components, by creating an entire independent components architecture, or use all these variants combined. For example, we can use the Web servers together with the application server, and just the database server on one separate machine.

The best solution is use the variant with all the components on separate machines using LPARs, which allows you to:

- Be more scalable about changing the configurations of the LPARs or adding new machines.
- Easily do administration and problem determination, where each component is responsible for one server, which means that only one component can generate error events.
- Easily implement high availability on the components, because each one is in its own server.

- Allocate the right number of processors and memory to each server, changing these values when your environment changes too.

Figure 3-11 shows the variant with all the components on separate servers.

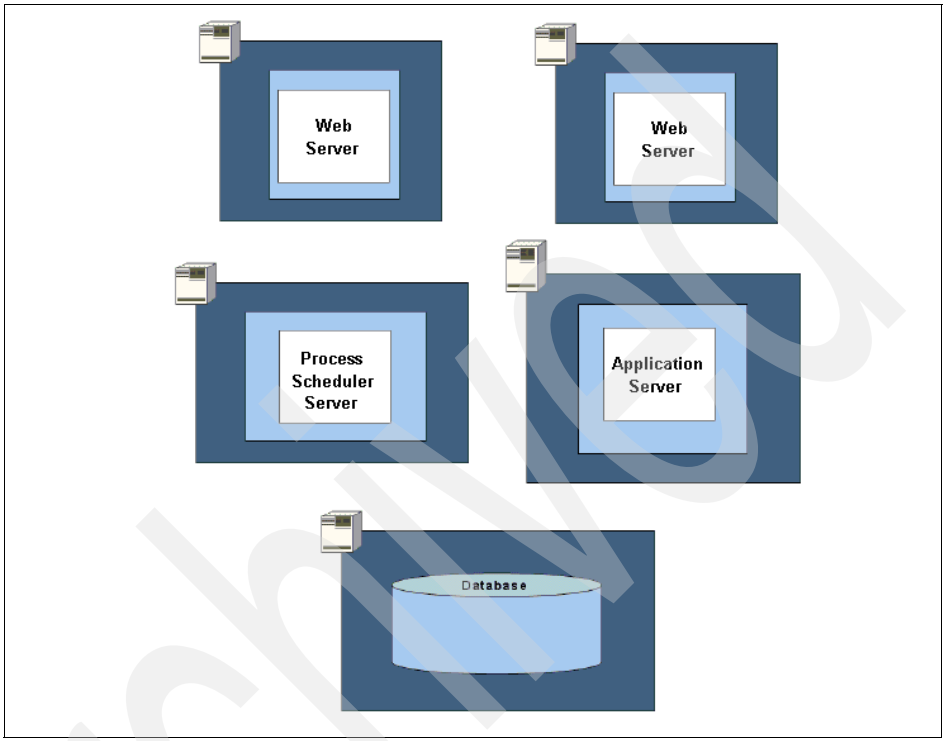


Figure 3-11 Variant with all the components on separate servers

3.4 Database planning

In this section, we discuss the high availability databases, including the possibilities for SAP installations.

3.4.1 Database HA options for SAP

SAP was developed to work with almost all RDBMS on the market, at least those that can support large databases, such as:

- IBM DB2
- Oracle Database
- Microsoft SQL Server™
- MAX DB

Every database vendor has its own technology, but in this section, we will show you some common HA solutions. We will review two approaches that have different alternatives and features, depending on the RDBMS software. We will also see how to combine these two approaches into one solution.

3.4.2 HA with only one database

This approach duplicates the shared storage so that both cluster nodes can read from the same disks (see Figure 3-12).

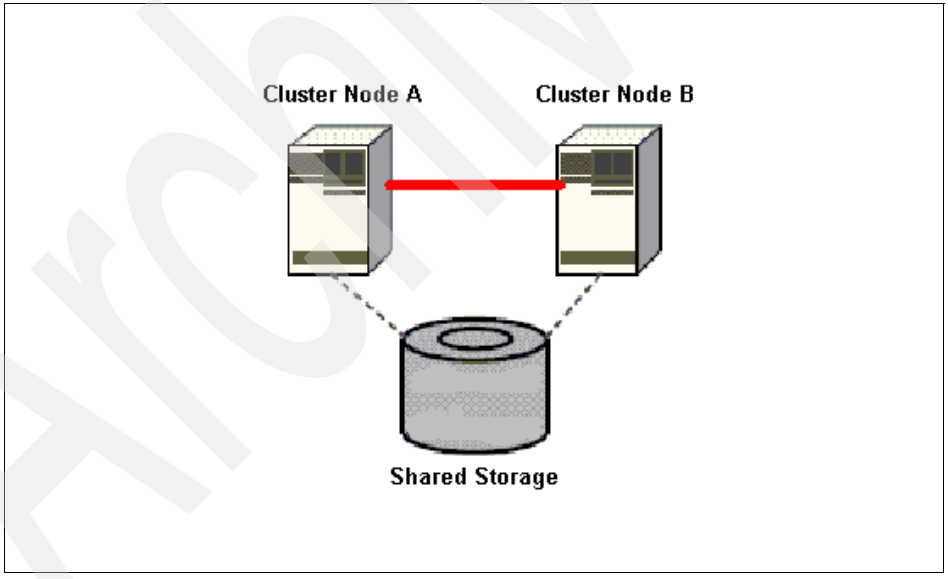


Figure 3-12 HA with one database

This technique has two alternatives: One is called the *fail-over cluster*, where each node has its own resources and, in the case of failure, re-assigns these resources to the standby node and restarts the database.

The failover might take some time because many steps must be taken to start the database on the standby node in order to guarantee the reliability of the data. These steps include:

- Detect the error.
- Change the network configuration.
- Check and mount the file systems on the standby node.
- Start the database with the crash recovery option on, to rollback all transactions that were already open and did not finish before the crash.

Attention: It is not possible to used load balancing on failover clusters.

The other alternative is often called the *parallel database*, where everything is shared between the cluster nodes, so each node can write “simultaneously”. Using this alternative, the cluster’s nodes will need the Distributed Lock Management (DLM) feature to manage the locks between the databases and will provide availability, scalability, and performance (load balancing) in addition to fast switchover.

3.4.3 Failover cluster versus parallel database

These two alternatives are quite different, as shown in Table 3-2.

Table 3-2 Failover versus parallel databases

Failover	Parallel DB
Solutions for all DB platforms.	Fast and transparent failover.
Good cache hit rate.	Horizontal scalability.
No performance impact after failover.	Load balancing between nodes.
Failover requires DB restart/recovery.	Uses all nodes at same time.
Long failover times.	Only two DB platforms.
Only one node works at time.	Overhead due to DLM.
Bigger hardware (vertical scalability).	Higher quantity of servers (horizontal scalability)

3.4.4 HA database with two databases

You have two alternatives when you have a HA database that uses two databases. One of them replicates the files on the database level and the other alternative replicates on the storage level.

The replication on the *database level* is maintained using a second database as a standby database to receive all log files by asynchronous replication. This solution is provided by different products with specific features for each database platform.

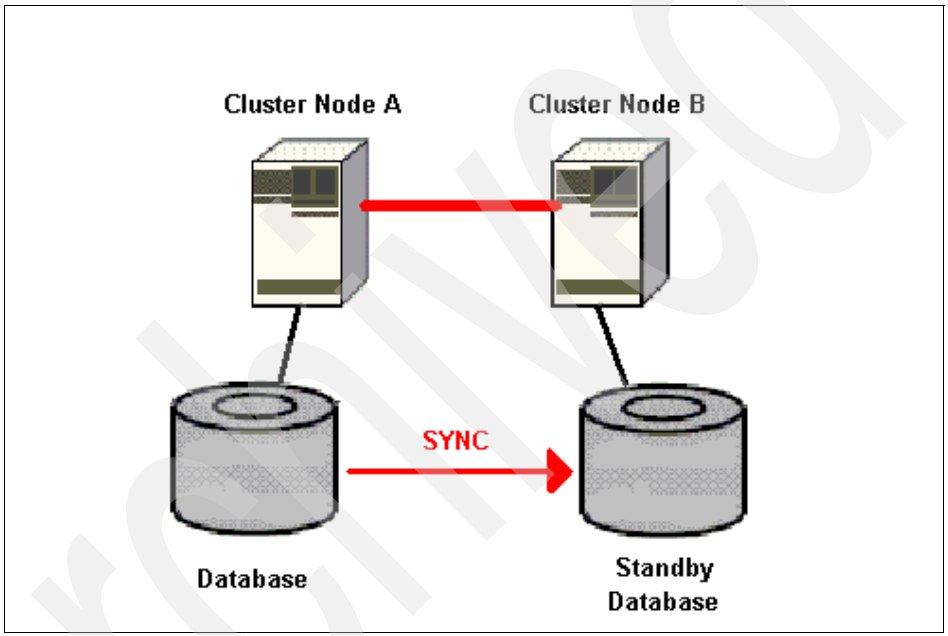


Figure 3-13 HA with two databases

This alternative requires that the standby database be in recovery mode in order to apply all archive logs that were shipped from the original database. This is the common solution used for disaster recovery solutions, but the asynchronous replication might lead to data loss on the disaster recovery site.

The replication on the *storage level* also uses replication, but the replication is executed by the storage solution. Using RAID (Mirroring), the storage can write on both disks at the same time. This solution provides an fast failover from the backup database to original one. This is different from the replication on the database level, because this solution is synchronous, and requires special and expensive special storage solutions to implement.

3.4.5 Clustering versus replication

Table 3-3 shows the benefits of these two solutions.

Table 3-3 Clustering versus replication

Cluster	Replication
Solution for local failover	An easy solution that enables a disaster recovery site.
Less disk space	Protection from logical errors when using asynchronous replication.
Transparent failover using a parallel database	Standby DB for system copies to test systems.
Load balancing on the parallel database solution	Geographic protection with remote replica

If possible, the best solution will combine these two solutions into one, in order to achieve a maximum level of availability with the benefits of both technologies, as shown in Figure 3-14.

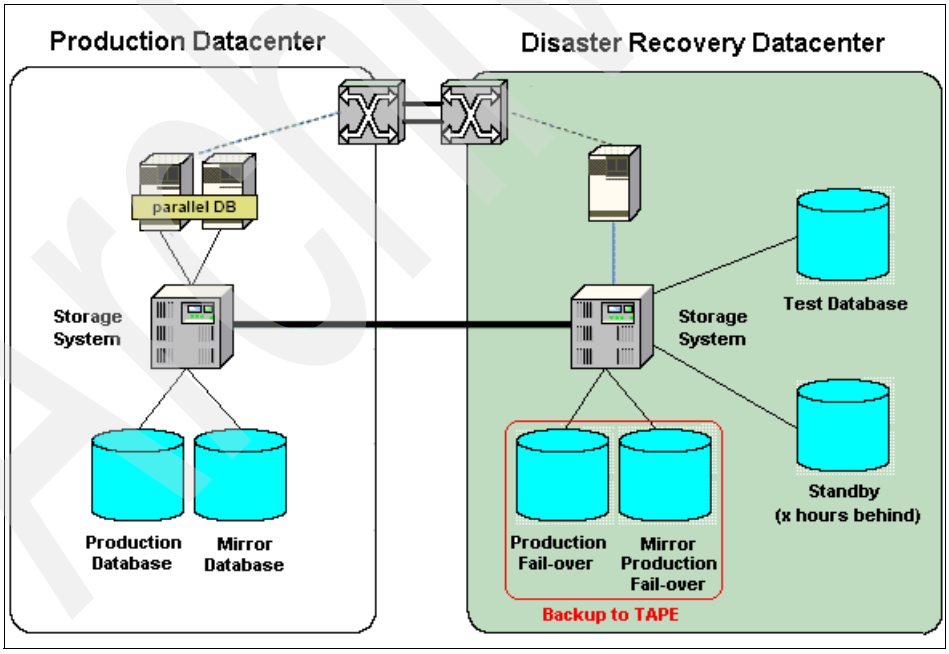


Figure 3-14 Combination of clustering and replication

Using this combination of technologies, we can have fast and transparent local failover and a disaster recovery site with synchronous storage replication or create test databases and standby databases to avoid logical errors (for example, table drops).

3.5 Network planning

In this section, we discuss the technologies to create high availability for our network environment while considering all the network components and the architecture level.

First, we need to split the networks up to understand the different environments and to improve high availability on each one.

- Server network
- Access network
- Company network

The *server network* connects all the application servers with the database server and other component servers of the solution, which means that it is very important to system availability, because if this network fails, the applications will lose their connection to the database and the entire system will be unavailable.

The *access network* is responsible for connecting the company network to the application servers. If it fails, the users that are using this network to access the application servers will be unable to access them.

The *company network* is the network to which the system front ends, printers, and other components belong.

Figure 3-15 shows the three layers of the network.

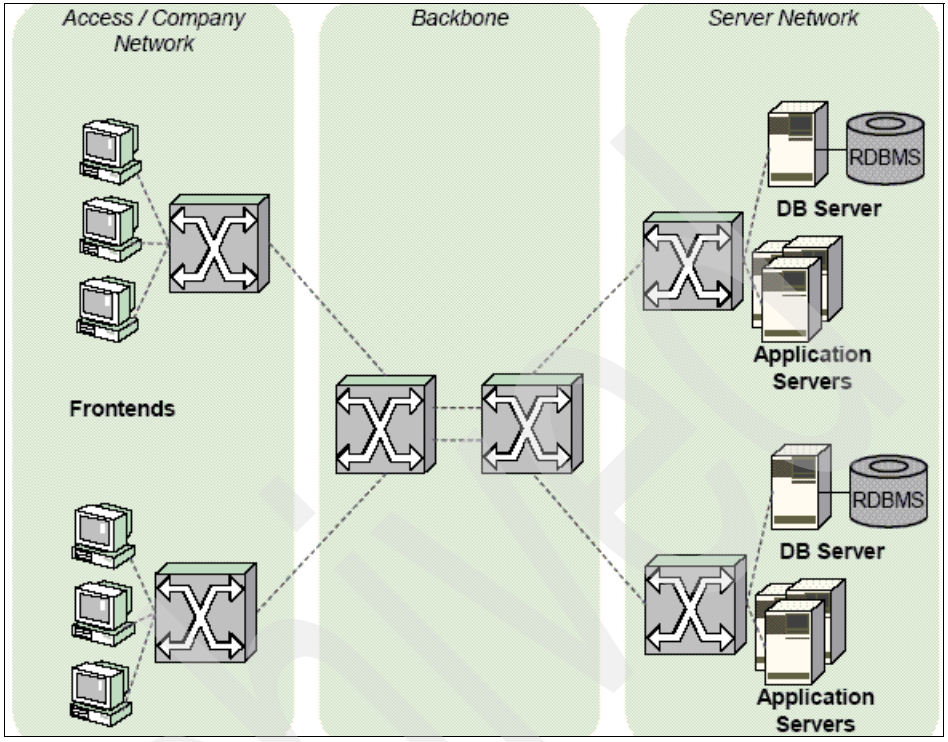


Figure 3-15 Three layers of the network

To create an high availability solution for this kind of network, we need to provide HA for all communication layers, from the physical layer to the transport layer.

The physical layer can be divided into two parts: The *network infrastructure* includes the cable, hubs, switches, and routers, and the *Network Interface Card (NIC)*, which every client and server in this network has, regardless of the technologies involved.

3.5.1 Network infrastructure and NIC

The network infrastructure needs to have an alternative network path between the application servers and the client applications, so that if the primary path connection fails for some reason, the network components can reroute the network traffic to the other valid path.

A good solution for this scenario is to use the Fiber Distributed Data Interface (FDDI) between the servers and Ethernet and to use redundant switches in the company network to create a infrastructure without a single point of failure.

If both technologies are implemented in the network infrastructure, the NIC still remains as a single point of failure. So, to correct this, redundant network cards must be installed to use the features of the redundant switches. This standby NIC is faster than switching the entire machine to a backup server.

Another solution is to use the proprietary switchover software or create your own scripts to switch the address of the failing NIC to the remaining NIC.

3.5.2 Transport and network layers

These two layers must be able to find alternative routes through the network by using an algorithm.

The *transport layer* is responsible for the end-to-end connection, and uses a sophisticated algorithm for retransmission and error detection based on confirmation packages and window size, so TCP need to be sure that a failure has occurred in the network in order to start resending the packages. This development is used to handle Wide Area Networks (WAN).

The *network layer* is responsible for finding routes and sending information to the next hop in the network. This routes are stored statically on each host as the gateway value, but you could set two gateways for each host; if one of them has a problem, the host has an alternative path for that package. You could also implement special routing protocols that can be used to “discover” routes using the information of other network nodes to find other routes for those packages.

Attention: Depending on the network architecture, the TCP timeouts are very short (especially Microsoft Windows Networks) and can be increased to avoid restarting communications on short network failures.

3.5.3 Access and server network

We recommend that each server on the *server network* have redundant network cards, so when a failure occurs, a switchover is performed to another network card, including the transfer of the IP address, as shown in Figure 3-16. These features are already incorporated into most of the available OSes. These network cards can be on different network segments in order to survive a crash of other network components.

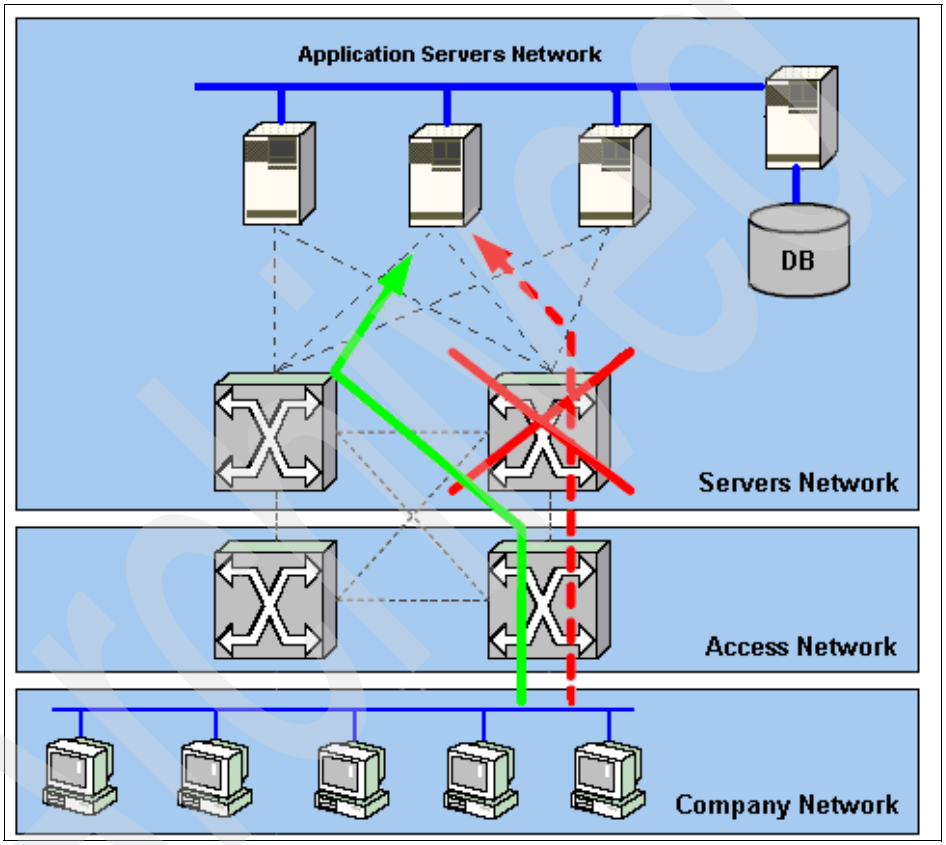


Figure 3-16 Network Infrastructure

In some cases, we recommend that you create a new network for the database server for storage that is separate from the database server for the application servers, and avoid using this one to execute the backup. The backup action can impact the performance of the LAN between the application and the database and degrade the response time of the application.

4

High availability solutions

"Who wants low availability systems?"[1]

High availability (HA) can be seen as a strategy to minimize system outage caused by hardware or software failure.

The sections in this chapter concentrate on the need for high availability in modern IT environments. We will explain the term "high availability" and discuss a few possible scenarios focused on Linux on POWER.

In 4.1, "High availability definition and background" on page 56, we discuss the term "high availability" in general and provide a number of definitions.

In 4.2, "HA scenarios and examples" on page 58, we introduce high availability scenarios and considerations using a hypothetical environment.

In 4.3, "Disk storage and high availability" on page 66, we show other pertinent features that are offered by Linux on POWER together with the IBM System p architecture in order to make your systems bullet proof and highly available.

Finally,4.4, "HA solutions" on page 73 concentrates on "ready to use" available HA solutions and components in a Linux on POWER environment.

Information an HA terminology is given in Appendix A, "High availability terminology" on page 93.

[1] Alan Robertson, driving force behind the Linux-HA project

4.1 High availability definition and background

During the process of designing and implementing the environment for mission critical applications, a number of questions are very likely discussed:

- Is the application or the service that will run on a piece of hardware important? If yes, how important? In other words, are you going to lose money if the application or the service fails?
- Who depends on this application or the service? For example, a failing LDAP server may affect hundreds of workstations that will be unable to authenticate.
- What happens if an important component of the hardware fails? For example, does the hardware provide redundant power supplies?
- What happens if the hardware fails completely?
- What happens if the complete site or building becomes unavailable to the users of the system?

If one of these questions is important for the environment you are setting up, or if you suspect that one of the applications might fail because of OS issues, you might want to consider setting up a high availability solution.

4.1.1 A definition of high availability

High availability (HA) refers to the availability of hardware and software (resources) in an IT environment. High availability can be achieved in a number of ways, ranging from the utilization of special redundant hardware to the use of off the shelf hardware in conjunction with software solutions.

Using special redundant hardware will normally lead to high availability, but the resulting overall costs are high. Thus, using off the shelf components together with software in order to achieve high availability is the more popular way. Both approaches are capable of helping a system survive failures.

A more formal negotiable definition of high availability has been defined by the IBM HA center of competence in Poughkeepsie, NY.

High availability Designed to provide service during defined periods, at acceptable or agreed upon levels, and masks unplanned outages from users. It employs Fault Tolerance, Automated Failure Detection, Recovery, Bypass Reconfiguration, Testing, and Problem and Change Management

Continuous Operations (CO)

Designed to continuously operate and mask planned outages from users. It employs nondisruptive hardware and software changes, nondisruptive configuration, and software coexistence.

Continuous Availability

Designed to deliver nondisruptive service to the user 7 days a week, 24 hours a day (there are no planned or unplanned outages).

If we look at the definitions ranging from high availability to continuous availability, we see that the level of high availability is directly dependant on the time a system or an operating environment is available for use.

Note: A typical IT operating environment consists of, but is not limited to hardware, software, data, environmental support, such as power and air conditioning, and processes.

A relative simple formula is often used to calculate this availability:

A [%] = MTBF / (MTBF + MTTR)*100

where MTBF is the mean time between failures and MTTR is the mean time to recover to operational status. The formula shows that an increasing MTBF tends to produce high availability and a decreasing MTTR also has a positive impact on the availability. If the MTTR tends towards zero, or in other words, the system recovers very quickly, the system becomes highly available. The recovery time includes the time to detect and analyze the fault and the time to reestablish the service. Thus, every high availability solution tries to minimize the recovery.

But we have to keep in mind the MBTBF value is a mere statistical value. An LED, a light bulb, or even a CPU may have an MTBF of 500,000 hours, but that does not mean it fails after 57 years of operation. It can fail at any time, and because IT environments consist of a high number of components, perhaps each of them with a different MTBF, the system availability based on MTBF values is very difficult to predict.

Because the MTBF of a complex environment cannot be calculated, high availability solutions generally try to minimize the MTTR. If MTTR is zero, the system is considered highly available, no matter what MTBF value enters the calculation.

After all, the availability of a system should always be seen from a potential user's viewpoint. The user uses the system to do his work and a user normally does not care about unexpected outages. And the user has a view of the overall system; if the hardware and software have an availability of 99.999%, but the network reaches a maximum availability of 80%, the system is more or less unavailable from the point of view of the user.

4.1.2 Common terms used in HA solutions

There are a few terms commonly used in high availability discussions that require some explanation:

Cluster In an HA environment, a cluster describes a collection of computer nodes and resources, such as disk storage and network connections. These resources cooperate and work together to provide high availability of services executed within the cluster. In case a clustered component fails, the resources required to run the component are transferred to another member of the cluster.

Cluster standby Describes the basic high availability configuration. One member of the cluster is up and running providing the defined services. The other member of the cluster is in a standby mode (idle). In case of a problem, the standby cluster is activated and control is given to the standby cluster. This mode is often referred as cold standby. It requires some hardware redundancy.

Cluster takeover In this configuration, all the nodes in the cluster are up and running. For example, one cluster executes critical applications, while other clusters perform non-critical tasks. In case of a failure, those clusters are able to take over theoretical applications.

4.2 HA scenarios and examples

There are many reasons for implementing HA solutions, which we will discuss in the following sections.

4.2.1 Possible failures requiring an HA solution

When Linux runs on POWER, it runs either on a single dedicated server or more often on a server divided into multiple LPARs. Looking at the latter option, we are dealing with an additional layer (the LPAR) and could have the following problems when running a system under Linux on POWER, which are generally called “single point of failure”.

- The IBM System p hardware or part of it may fail or it could fall into multiple non-recoverable errors causing the whole server to fail.
- The disk subsystem could fail.
- The code controlling the virtualization could fail.
- Linux could fail.
- The application running under Linux could fail.

The possibilities of one of these problems occurring is different. A complete failure of a System p server is rather unlikely, while running into non-recoverable problems with an application is most likely. The other single points of failure are somewhat in between these two extremes. Implementing high availability is the art of eliminating existing single points of failure.

So, the best way to eliminate single points of failure is the duplication of critical resources. Duplicating a resource is a somewhat generic term. It could be the duplication of the complete hardware environment and may lead to very high costs. On the other hand, duplicating a software environment or an application can only eliminate a single point of failure and produce much lower costs, as shown in Table 4-1.

Table 4-1 Single points of failure

Single point of failure	Probability	Cost to fix
System p hardware	Low	High
Disk Subsystem	Low	Medium
LPAR	Low	Low
Linux	Low	Very low
Application	High	Very low

Table 4-1 on page 59 lists the single points of failure just discussed and their probabilities and possible costs. Also, there are some other reasons that may lead to an application or service becoming unavailable. There can be scheduled maintenance, a required reconfiguration of an LPAR that might require a reboot or a required Linux patch requiring reboot. However, all these possible outages are under complete control of the staff running the environment.

4.2.2 Application environment examples

In this section, we will discuss a few typical application examples and analyze their need for high availability. Also, we will sketch out how we achieve the required level of high availability in an POWER environment.

No HA required

We start with a simple but typical modern application outlined in Figure 4-1. All the applications reside on a dedicated piece of hardware.

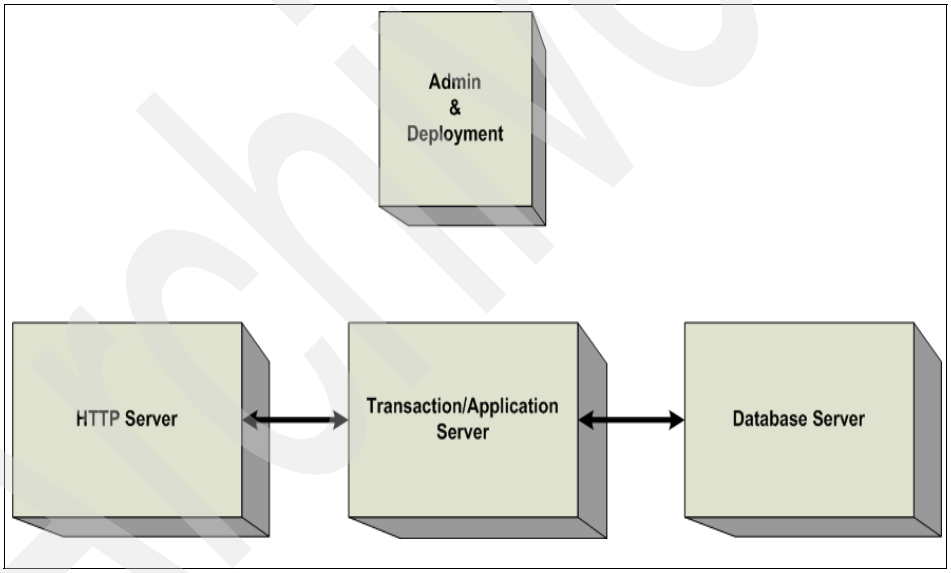


Figure 4-1 Simple application with no HA required

- A HTTP server acts as a front end and accepts requests from clients using HTTP transactions to communicate.
- A WebSphere® Application Server hosts the actual customer application.
- A DB2 database server serves as the database back end.
- A WebSphere deployment manager for administration purposes.

All these components reside on dedicated servers sized for the application.

By analyzing this environment for single points of failure, we will find that:

- The hardware may fail. This applies to any server in the environment.
- Linux may fail. This applies to all servers in the environment.
- Any application may fail. This too applies to all the servers in the environment.

Most likely, an application can fail and must be restarted, which will take a few minutes, or Linux needs to be rebooted. In this case, the downtime can be in the range of five to seven minutes. If this time frame is acceptable, no further activity is required; if not, an environment providing high availability should be implemented.

High availability required

If an extended downtime caused by a failing component is not tolerable, high availability should be implemented if the additional cost is acceptable.

Figure 4-2 on page 62 shows the same scenario as in Figure 4-1 on page 60. We see the same components as in our initial scenario with some significant changes:

- All the initial components are duplicated.
- The HTTP front-end and transaction servers have additional connection paths so that every unit may communicate to the other units.
- The database server is also duplicated, but uses database replication as the standard backup method.

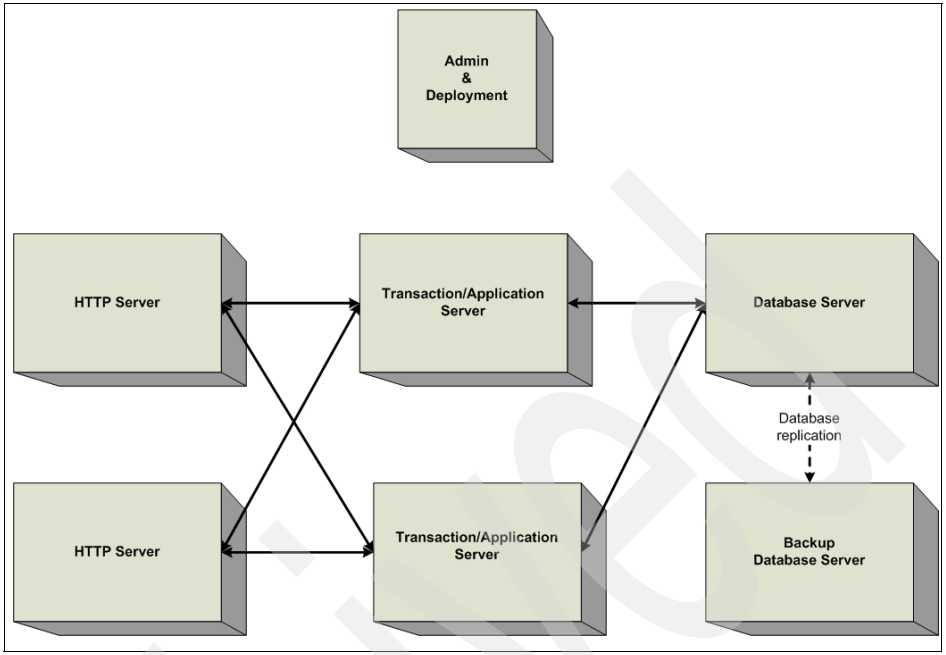


Figure 4-2 Duplication of components

The result is clear: If one of the components crash, the spare component takes over. Using this solution, we eliminated the single points of failure mentioned in 4.2.1, “Possible failures requiring an HA solution” on page 59. Consequently, each of the “building blocks” of our now highly available system requires a dedicated piece of hardware. In normal environments, this would be a dedicated server and additional resources like shared storage and so on, which can lead to a very expensive solution. Also, the effort to administer and run such an environment would be much higher.

Duplicating resources and implementing control to monitor outages and provide an immediate takeover is the most expensive solution and may therefore not be the best solution.

Fortunately, the POWER architecture features a virtualization engine, which allows you to virtualize complete server environments in a partition, called LPAR, inside of a single POWER server (see 1.4.1, “Running Linux on POWER servers” on page 12). Beside the virtualization of CPU, BIOS, and memory, POWER virtualization delivers network connectivity in the form of a virtual Ethernet type network adapter and an interface to disk storage in the form of an virtual SCSI adapter. Instead of distributing the environment to (at least) six servers, we can use the virtualization capabilities of the POWER platform, as shown in Figure 4-3

on page 63. The applications are still duplicated, but reside in six separate LPARS on a single POWER server.

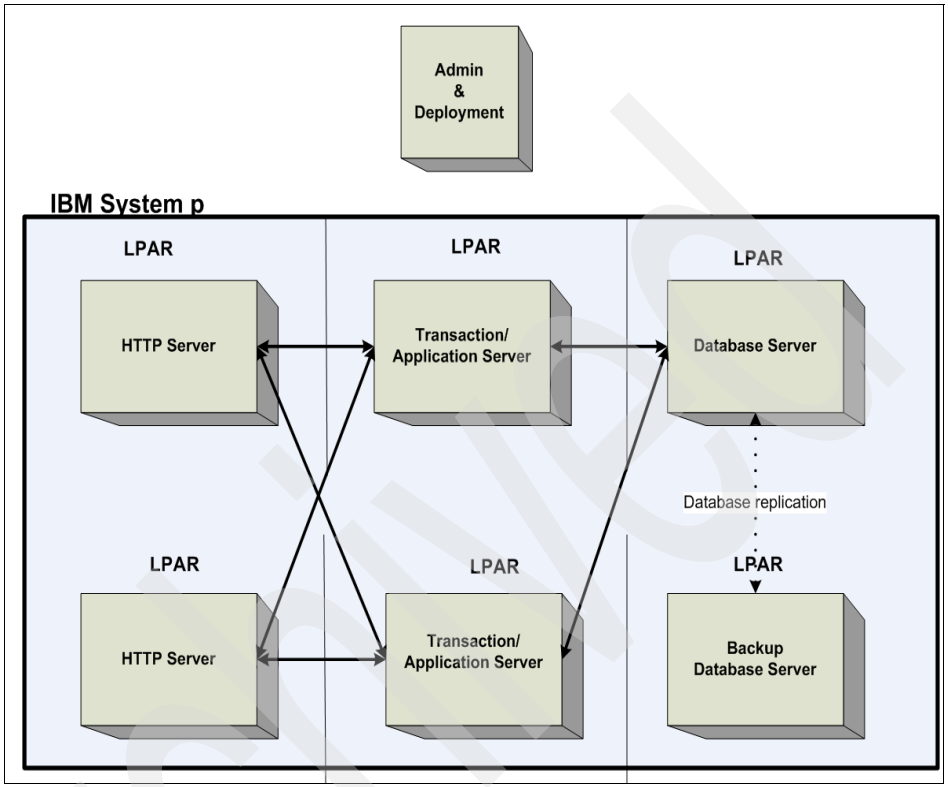


Figure 4-3 Duplicate components on a single POWER server

Analyzing this configuration, we eliminated a number of single points of failure compared to the initial non-HA solution in Figure 4-1 on page 60. A single point of failure exists in this configuration: if the hardware fails completely, the application would be unavailable. So, what we really need is the elimination of a possible outage caused by failing hardware. In addition, we want to avoid the complexity and the high costs when we deploy the components onto dedicated servers.

Figure 4-4 shows a good mix between cost and complexity while eliminating the single point of failure caused by failing hardware. By deploying two sets of components to two POWER servers providing the required LPARS, we finally eliminate the possible fault caused by failing hardware.

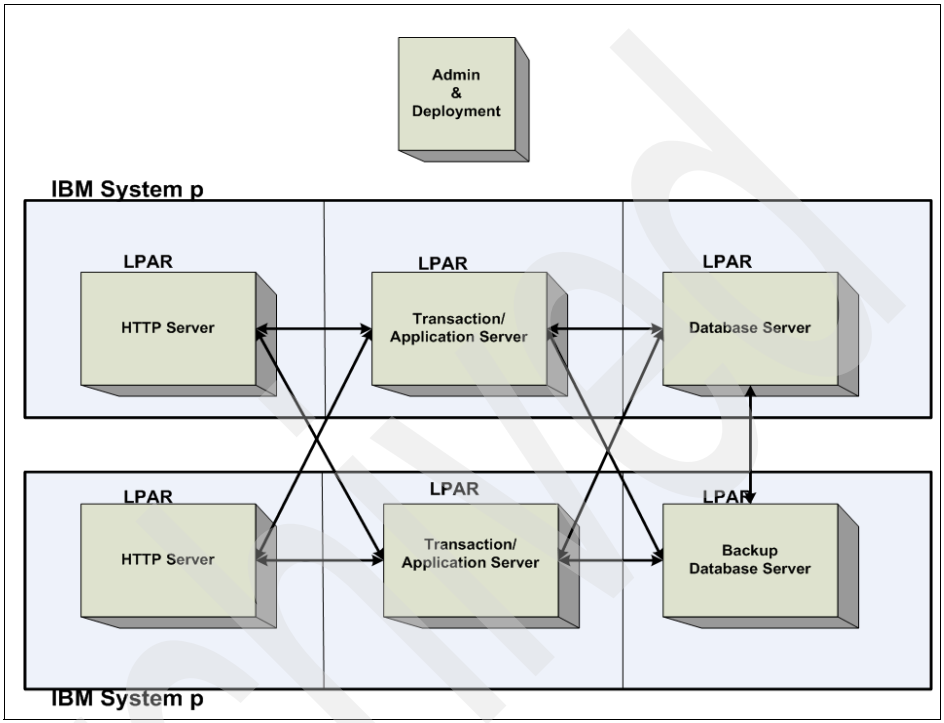


Figure 4-4 Using two POWER servers for HA

4.2.3 A configuration example

Consider the system configuration shown in Figure 4-5 on page 65.

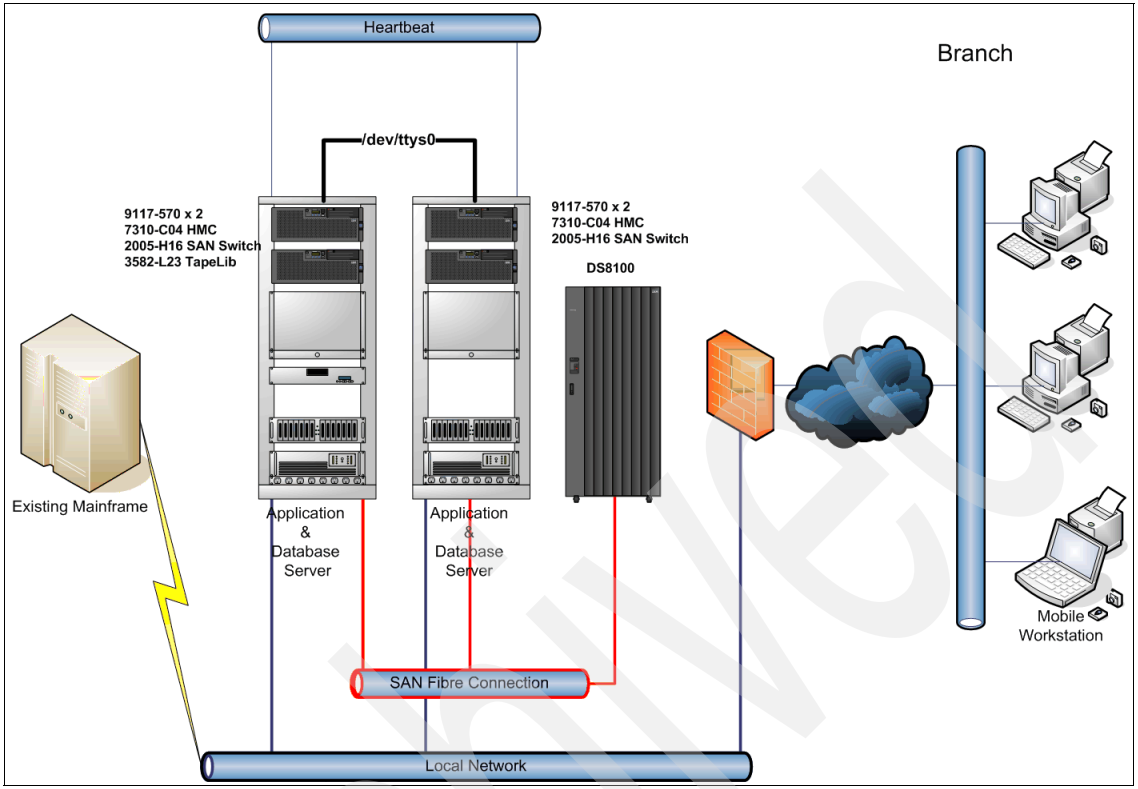


Figure 4-5 A typical HA environment

This system configuration presents a snapshot from a actual system built to provide central support for document issuing purposes, such as issuing a driver's license or a passport. There was no discussion about the requirement to have the system always available during business hours. The configuration was made highly available by proposing the following configuration:

- The application and database servers are put on separate IBM System p5 570 machines.
- Both the application and database servers are running redundant active/passive configurations. That means the backup system is on standby and activates in case of an error.
- One of the redundant configurations has a tape library for backup purposes. A tape library is important for well defined systems management, but in this case it is not mission critical, so it is not made redundant.
- The HA communication (Heartbeat) uses dedicated communication lines, that is, a separate network segment and a serial line.

- Disk storage is attached by connecting a IBM TotalStorage 8100 System through a fiber based Storage Area Network.

For the application part, the system does not have single points of failure. You may notice the single storage array. How are they made highly available? We will discuss this topic in 4.3, “Disk storage and high availability” on page 66.

The external components, such as the firewall, the network connection to the clients, and the mainframe back end are redundant.

4.3 Disk storage and high availability

Eliminating single points of failure by introducing redundancy on the server, or in the case of Linux on POWER on the server and LPAR level, is only the first part of making an application highly available.

So far, we have discussed the more visible parts, that is, eliminating the single points of failure by duplicating our operating environment. While this keeps the operating environment available, we did not discuss how to make the required data highly available as well.

Disk storage is no longer a limited resource and the price for a gigabyte of disk storage is continuously decreasing. With this decreasing cost per gigabyte, it makes sense for a high availability design to have a look at the disk storage environment and discover how to eliminate single points of failure.

A hard disk is definitely a single point of failure, a painful experience that almost everybody working with a computer might have been experienced.

Also, almost everybody working with (personal) computers knows the term *disk partitioning*. Before the operating system is put on the disk, you are required to decide how much storage you are going to assign to the operating system and to your data storage. Under Linux, partitioning and the layout of the various file systems is a major planning issue, no matter what solutions the various installation programs provide you with regarding the layout based on the available resources.

With this in mind, we will now discuss how to handle these shortcomings.

4.3.1 An introduction to RAID

Disk performance and data security is often an issue in enterprise systems. Modern systems can compute data often faster than the existent I/O can store that data, which makes disk I/O crucial for overall performance. In addition, the generated data must be stored in a safe way and must survive a hardware failure, and therefore must be highly available. One solution to address these issues is called RAID. RAID stands for Redundant Array of Independent Disks. In other words, RAID is being used to store data across more than one physical disk for two essential reasons:

Redundancy Data is stored on more than one disk in a manner that allows the data to be recovered in case physical members of that group fail.

Performance Because more than one single disk is involved in the operation, more than one head is moved, which results in higher performance.

In general, RAID can be provided as a software solution (software RAID) or as part of the disk storage hardware itself (hardware RAID).

Linux on POWER offers both versions of RAID, so we will discuss both types.

4.3.2 RAID types

RAID itself offers a number of different implementations. Whether it is software or hardware RAID, the functions are the same. In this section, we will limit the discussion on RAID to RAID types 0, 5, and 10.

RAID 0

RAID 0 is often called data striping. It is primarily used for performance. Data is written to multiple disks in an array just to increase its performance. In addition, RAID 0 allows the combination of a number of disks into large logical volumes.

Figure 4-6 shows a RAID 0 layout using three physical disks. The layout is in the form of alternate stripes.

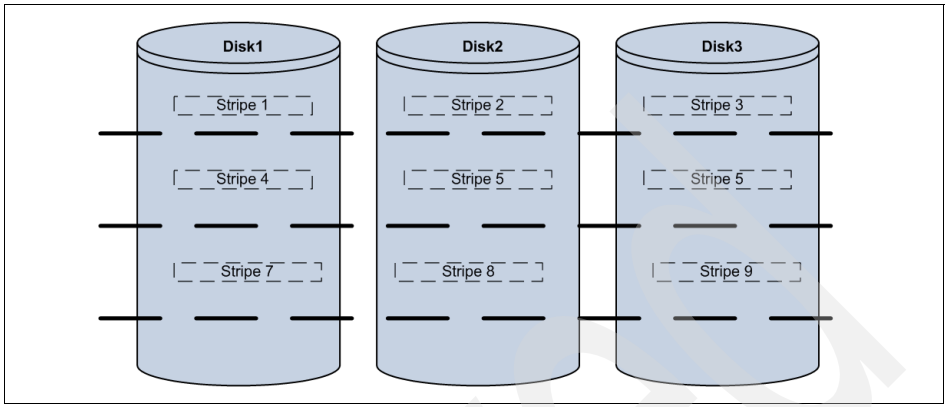


Figure 4-6 RAID 0 striping

The definitive drawback of RAID 0 is that it just offers a performance increase but lacks any security. If one disk fails, the complete volume is considered damaged. For our discussion on high availability, RAID 0 is introduced for informational purposes only.

RAID 5

RAID 5 adds data security to increased performance. It still uses a stripe layout across multiple disks, but it uses stripes to write parity data, as shown in Figure 4-7.

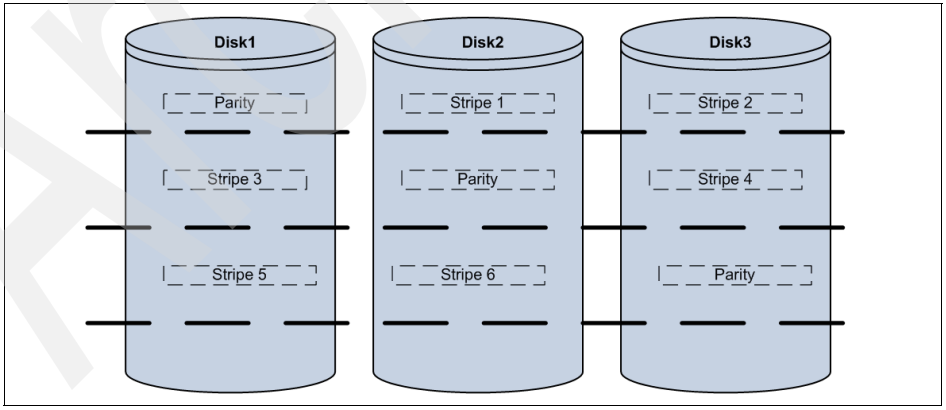


Figure 4-7 RAID 5 layout

In case a disk fails, the parity data can be used to recover the missing data, thus eliminating a single point of failure. The performance compared to a single disk is still good with the penalty of a decreased disk space because of the set aside parity stripes, as shown in the following equation:

(capacity = SizeOfDisk*(NoOfDisks-1))

This will eliminate a single point of failure, but the recovery of data out of the parity information is a somewhat time-intensive process. In this sense, RAID 5 provides increased security but does not help keep the application available for the user in case a disk fails.

RAID 10

RAID 10 uses the stripe layout found in RAID 0 and RAID 5, but adds mirroring to the picture, as shown in Figure 4-8. That means RAID 10 disks are always assigned in pairs. In the case of disk failure, no data loss and only a slightly reduced throughput would be the result, which makes RAID 10 the favorite but most cost intensive disk layout in high availability environments.

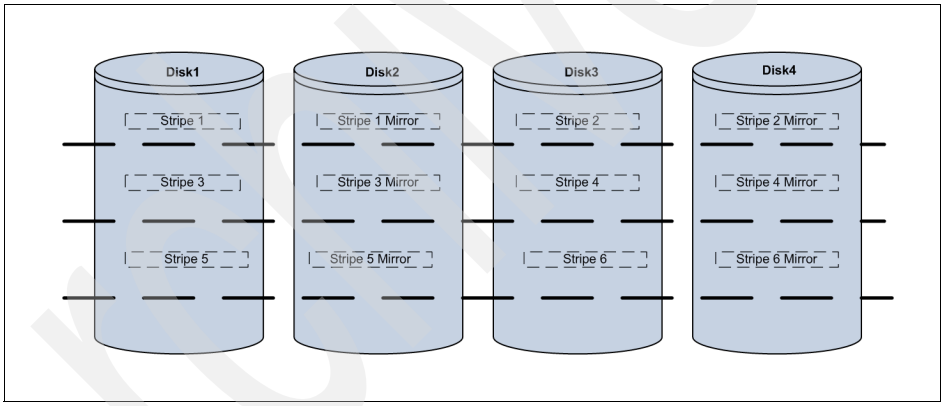


Figure 4-8 RAID 10 layout

Summary

Table 4-2 summarizes the RAID factors discussed. The capacity is an approximate value that may slightly vary depending on the implementation.

Table 4-2 RAID type summary

RAID type	Redundancy	Disk capacity	Read performance	Write performance
RAID 0 (Striping)	None	100%	Very good	Excellent
RAID 5 (Striping+Parity)	Very good	67% to 90%	Very good	Good
RAID 10	Excellent	50%	Excellent	Very good

4.3.3 RAID and Linux on POWER

In general, both software and hardware RAID is widely supported under Linux on POWER. As far as software RAID is concerned, the installation and configuration may differ depending on the distribution, but the results are ultimately the same.

Software RAID

As its name implies, software RAID is a software implementation of RAID. It is widely used because of its lower cost and because it works with off the shelf hardware.

Most of the distributions provide you with reasonable tools to configure and maintain software RAID.

The most important thing to remember when it comes to software RAID is that it works with disk partitions as opposed to hardware RAID, which works with whole physical disks. Figure 4-9 on page 71 shows a layout of an generic software configuration under Linux.

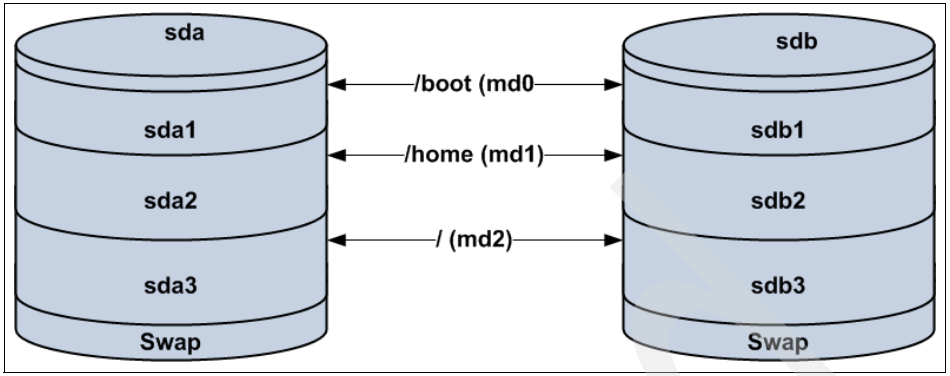


Figure 4-9 Software RAID under Linux configuration

There are two disks that have been partitioned exactly the same. The disks will contain the /boot, /home, and / (root) file systems as well as some swap space. In this setup, sda1 and sdb1 would be assigned as md0 and so on. This association is normally stored, along other pertinent information, under /etc/raidtab.

You may notice that swap space is not included in the RAID mechanism. Many sources about software RAID suggest that you do not include swap space in the RAID scheme. This recommendation may make software RAID less than ideal when it comes to high availability and mission critical environments. In any case, if swap operations are carried out during a drive failure, faults such as a system down situation caused by a kernel panic may occur.

Under Linux on POWER, a RAID configuration like the one shown in Figure 4-9 would not work. Software RAID requires a kernel module to work; Linux on POWER cannot boot from a software RAID system because it does not have access to the /etc file system and its /etc/raidtab file.

In general, when using Software RAID and Linux on POWER, you should be very familiar with the boot process and carefully select your disk layout. An alternative is shown in Figure 4-10. It uses software RAID on the data partitions only.

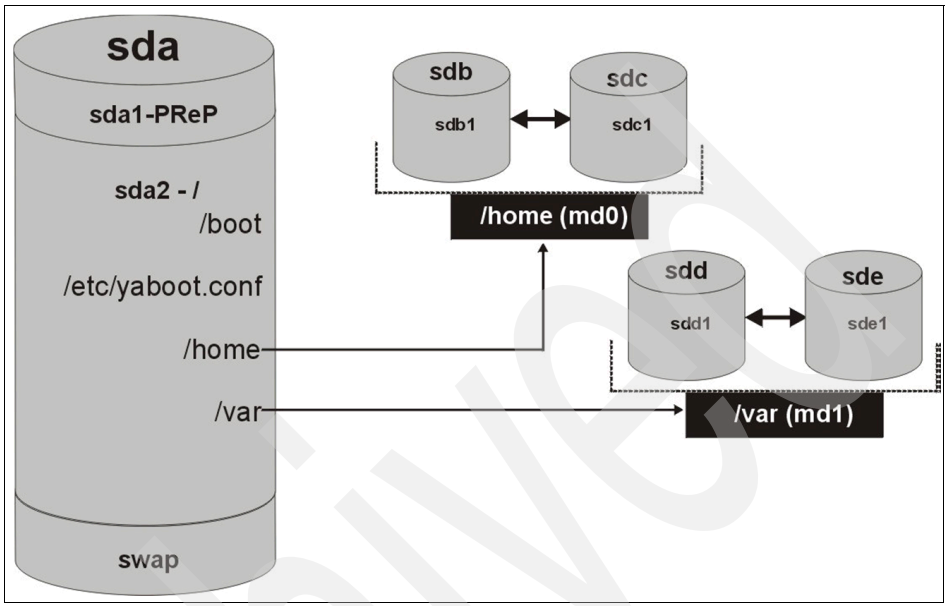


Figure 4-10 Example Linux on POWER software RAID layout

This is acceptable, even from an high availability viewpoint. In the case of a severe system failure that causes the hardware or the OS to reboot, a takeover would be initiated.

Hardware RAID

Hardware RAID is often a compelling solution for Enterprise computing and its disk I/O needs. One of its major advantages over software RAID is that no software is required by the operating system to recognize the disk arrays because the RAID function occurs at the hardware level. The real motivation for hardware RAID, however, comes down to performance and redundancy.

Hardware RAID provides performance by allowing these disk arrays to work in unison “as one disk.” Unlike software RAID, which uses CPU power to perform I/O operations, hardware RAID uses the processor provided with the I/O subsystem, whether it is a co-processor on an SCSI PCI card or a separate disk subsystem.

For all POWER5 systems, the RAID cards that support internal drives are all PCI-X adapters with the following features:

- Dual or Quad Channel, Ultra320 SCSI controllers that can produce up to 320 MBps speeds.
- Has a hardware XOR DMA Engine with a non-volatile write cache.
- Supports RAID levels 0, 5, and 10.
- Disk arrays are bootable.
- Supports external I/O devices like non-RAID disks, tape, and optical.

Table 4-3 lists describes the cards that can perform RAID on the internal disks for Linux on POWER at the time of this writing.

Table 4-3 Available SCSI pci Adapter for system p

Card Type	PCI Form Factor	Channels	RAID Levels Supported	Write Cache Size (MB)	Read Cache Size (MB)	Adapter FFC	Recharge Battery Technology	Cache Battery Pack Concurrent Replace?	Cache Battery Pack FFC
2780	Yes	4	0, 5, and 10	Up to 757 (comp)	Up to 1024 (comp)	2527	LiIon	Yes	2D01
5702	Yes	2	None	0	0	2522	None	N/A	N/A
5703	Yes	2	0, 5, and 10	40	0	2523	NimH	No	2526
5709	No	2	0, 5, and 10	40	0	2523	NiMH	No	2526

The big advantage over Software RAID is the transparency of your disk configuration, that is, you do not have to care about the placement of your essential file systems.

4.4 HA solutions

In this section, we will have a look at real solutions for high availability purposes on the Linux on POWER platform. We will start with an introduction of the basic components Linux itself provides.

4.4.1 Linux-HA

Driven by the Linux High availability project (http://www.linux-ha.org/), a scalable and complete high availability solution has been developed under the GPL. The Linux-HA project concentrates on providing the required software to implement high availability. However, other important requirements, such as shared file systems, are addressed somewhere else.

Linux-HA runs on all supported Linux platforms including Linux on POWER.

Linux-HA Version 1

The central program provided by Linux-Ha Version 1 is called *Heartbeat*.

Heartbeat provides the basic functionality to build a two node high availability system. Heartbeat is able to detect the failure of a node. However, for a complete high availability solution, Heartbeat requires other components to implement failover functionality:

- An alert management tool and monitoring tool. In most cases, **mon** is used. **mon** is a general purpose tool to monitor services and alert you in the case of detected failures. It provides an open interface for both the monitor and the alerting part. Monitors and alerts can be implemented using a number of programming languages, such as C, Perl, and so on.
- A tool to monitor connectivity between two nodes. Under normal operation, Heartbeat will not initiate a failover because just a network interface becomes unavailable. **ipfail** watches special, so called ping nodes. If a ping node does not answer, **ipfail** will contact the other node of the cluster and ask if this member has better connectivity. If yes, an interface or a network connection became unavailable, which may lead to resource failover.

You can think of Heartbeat as an extension of the init process to a clustered environment. It makes sure its processes are up and running at all times and restarts them accordingly.

Communication between Heartbeat on different nodes can be configured in a number of ways. Basically, Heartbeat is able to use UDP communications from UDP unicast to UDP broadcast. In addition, it can communicate through serial lines. Figure 4-11 on page 75 shows an overview of the Linux-HA Version 1 architecture.

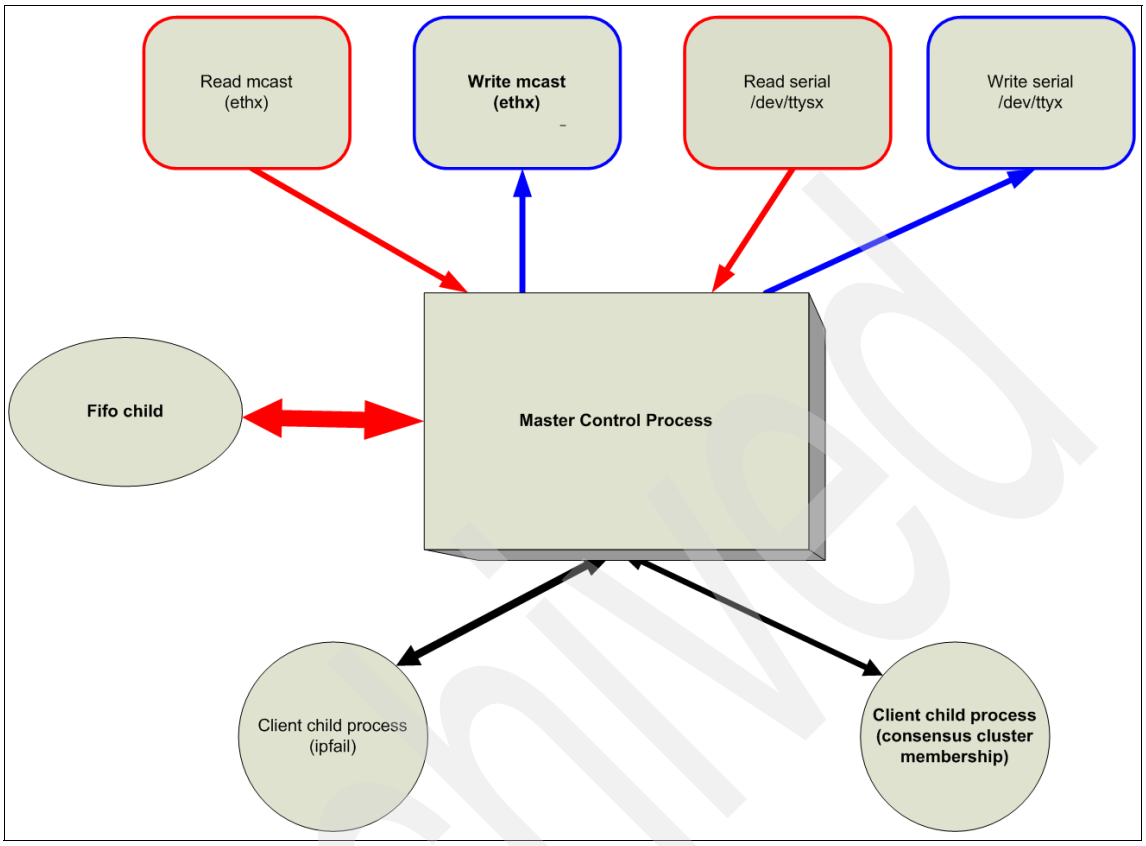


Figure 4-11 Linux HA version 1 architecture

In case Heartbeat does not receive a packet from the partner node, the node is considered dead and Heartbeat starts to fail over any services running on the other node. In addition, if STONITH is supported and configured, the failing node is shut down.

Note: STONITH stands for Shoot The Other Node In The Head, and it is a functionality to force a remote shutdown of a failing or stampeding node. Certain hardware is required to support this functionality. For example, the System Hardware Management Console provides STONITH support.

Heartbeat can also monitor routers and switches as though they were cluster members using the ping or ping_group directives. Combined with the **`ipfail`** program, Heartbeat can also cause failovers when networking connectivity is compromised.

Heartbeat provides services (which it calls resources) as configured in the one of of its configuration files, which are called *haresources*. The services, or resources in the haresources file, are started in a left-to-right order, and stopped in a right-to-left order.

Linux-HA Version 2

While Version 1 of Linux-HA was limited to a two node high availability setup, the new Linux-HA Version 2 now supports multi-node configurations. The architecture is now much more mature and features a modular design.[2]

Like other, non open source high availability solutions, Linux-HA maintains its relationships now by means of resources and resource groups (see “Linux-HA” on page 94)

Figure 4-12 gives an overview of Linux-HA Version 2.

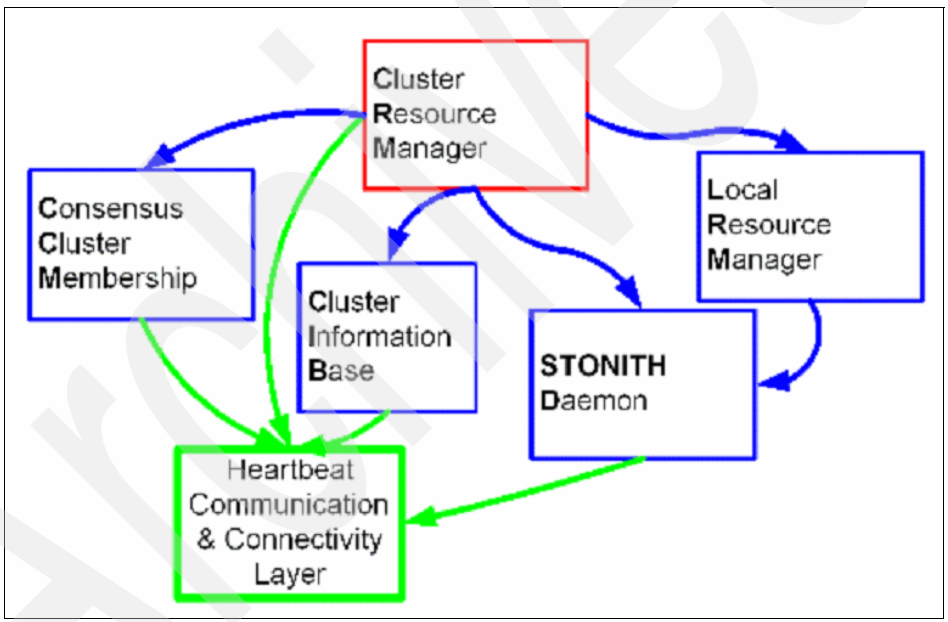


Figure 4-12 Linux-HA Version 2 overview

Its features include:

- Works on all known Linux variants and platforms, and has been shipped as part of a number of distributions.
- Has fast (<second) server failure detection.

[2] This information was taken from http://www.linux-ha.org.

- Has sophisticated knowledge of service inter-dependencies, and always starts and stops resources quickly and correctly.
- Supports both co-location and anti-co-location constraints.
- Policy-based resource placement puts services exactly where you want them based on dependencies, user-defined rules, and (user-defined) node attributes and location constraints.
- Fine-grained resource management can include user-defined attributes, meaning that failover can occur on any user-defined criteria.
- Time-based rules allow for different policies depending on time. This means that you can have different rules for where resources are located, and you can also set fallback policies differently by time. For example, you can say that fallback is normally delayed until night or weekends.
- Easy to understand and deploy init-based application management; most services require no application scripts for basic management.
- Resource groups provide simple, easy-to-use service management for straightforward environments.
- Active fencing mechanism (STONITH) provides strong data integrity guarantees, even in unusual failure cases.
- Has a full-featured GUI for configuring, controlling, and monitoring HA services and servers.
- Has Common Information Model (CIM) support for Systems Management support.
- Has a long history of Linux and strong reputation for robustness.
- Works on stand-alone systems to provide easy-to-use init-script-based service monitoring and restart.

Figure 4-13 on page 78 shows the Linux-HA Version 2 architecture in more detail. It consists now of a number dedicated modules.

- Heartbeat: The strongly authenticated communications module known from Version 1.
- CRM: The Cluster Resource Manager maintains the resource configuration, controls the execution of resources, and monitors the state of resources. The CRM interacts with every component in the system:
  - PE: The CRM Policy Engine computes a resource transition graph.
  - TE: The CRM Transition Engine executes PE computed transition graphs.
  - CIB: Cluster Information Base. A repository for cluster resource and node information as seen by CRM.

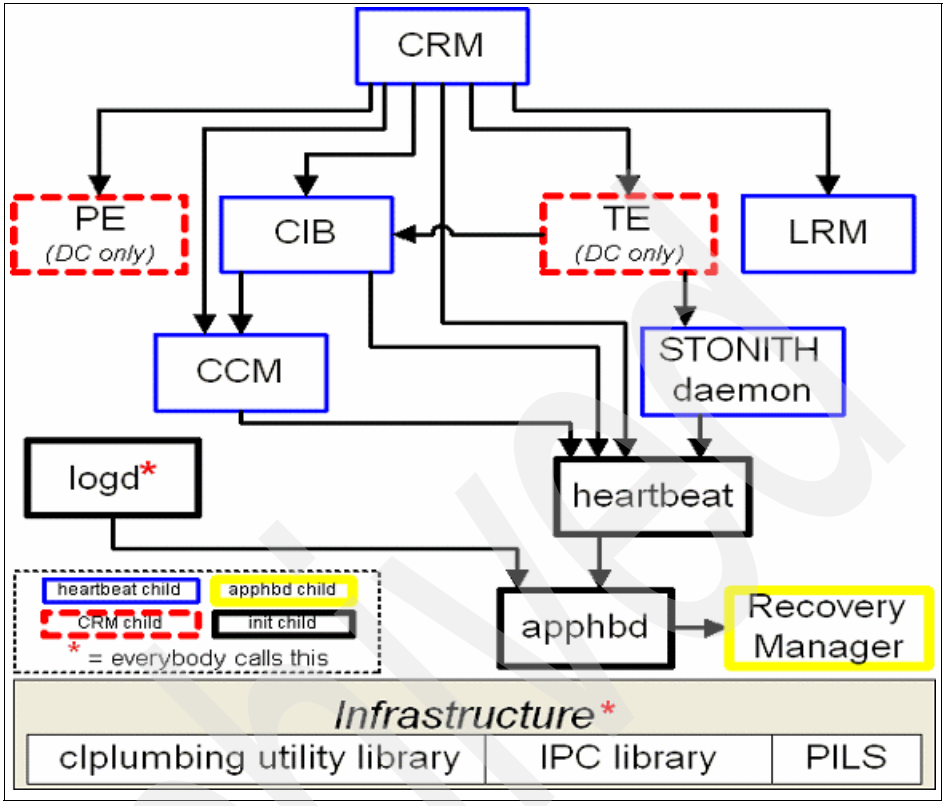


Figure 4-13 Linux-HA detailed architecture

- CCM: Strongly Connected Consensus Cluster Membership. Ensures that every node in a computed membership can talk to every other node in the same membership.
- LRM: Local Resource Manager. Start, stops, and monitors resources on behalf of the CRM.
- STONITH Daemon: Provides node reset services.
- logd: Non-blocking logging daemon.
- apphbd: Application level watchdog timer service.
- Recovery Manager: Application recovery service.
- Infrastructure.
  - PILS: Plug-in and Interface Loading System.
  - IPC: Interprocess Communication.
  - Cluster "plumbing" library.

– CTS: Cluster Testing System - cluster stress tests.

Shared storage

Linux-HA has no special shared storage requirements. It supports a number of data sharing configurations:

- No shared data.
- Replication, for example, though DRBD, which is a Linux block device that implements RAID 1 over a network.
- SCSI RAID controllers supporting clustering, for example, IBM ServeRAID™
- Any kind of external storage units.

The only requirement Linux-HA has on the shared storage subsystem is the ability to get attached and released using the **mount** and **unmount** commands.

4.4.2 Tivoli System Automation for Multiplatform

Tivoli System Automation for Multiplatform manages the availability of applications running in systems or clusters on various IBM platforms, including Linux on POWER, and offers the following features:

- High availability and resource monitoring
- Policy based automation
- Automatic recovery
- Automatic movement of applications
- Resource grouping

General product information can be obtained at the following Web page:

http://www.ibm.com/software/tivoli/products/sys-auto-linux/

A number of white papers as well as some policy examples are available at the following Web page:

http://www.ibm.com/software/tivoli/products/sys-auto-linux/downloads.html

A more detailed introduction than this overview can be found in *End-to-End Automation Management User's Guide and Reference*, SC33-8211

End-to-end Automation with IBM Tivoli System Automation for Multiplatforms, SG24-7117, which is an IBM Redbooks publication on Tivoli System Automation, is also available.

In an IT context, automation means that a desired runtime behavior of IT technology is defined in a formal way and executed by an automation engine on behalf of a human operator.

While this is true for most areas in operations management, Tivoli System Automation for Multiplatform concentrates on automating the availability of IT resources. In other words, it provides the framework for high availability solutions, as shown in Figure 4-14.

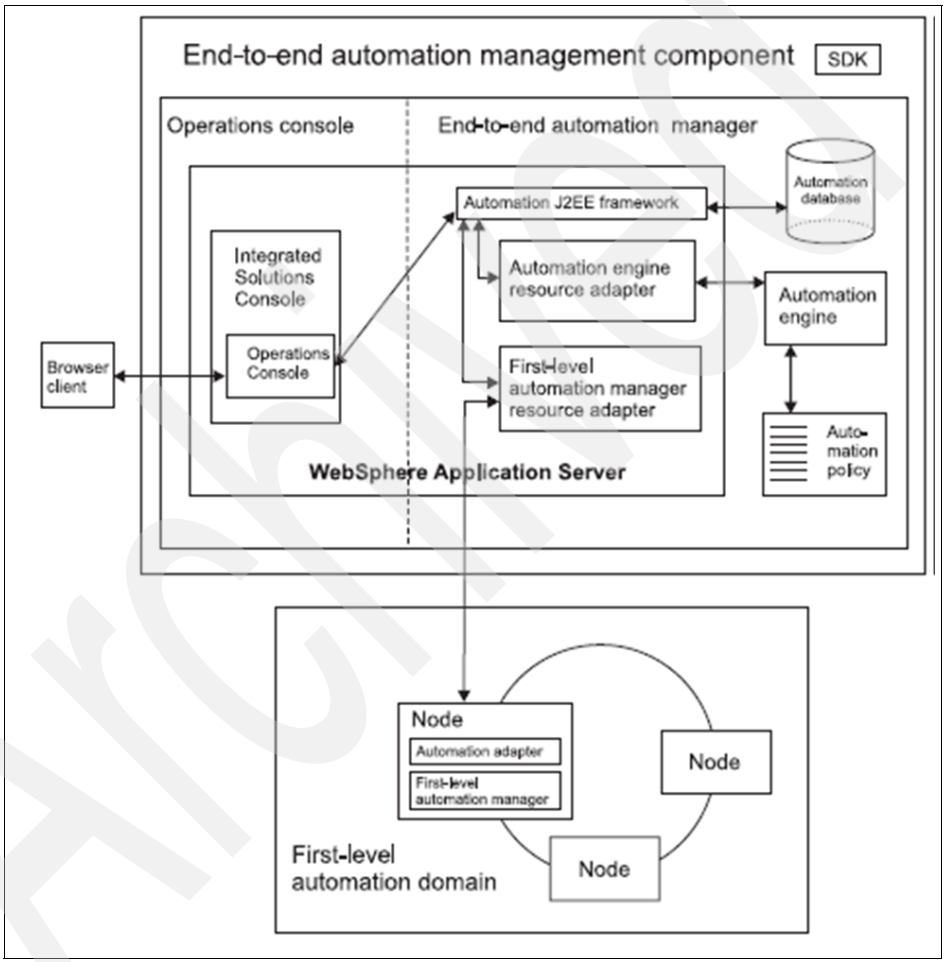


Figure 4-14 Tivoli System Automation components

Figure 4-14 show the general layout of Tivoli System Management for Multiplatforms:

- A standard HTML browser acts as the user interface.

- An end-to-end automation management component runs in a WebSphere Application Server and provides the automation infrastructure. It contains console interfaces, the automation engine itself, and a repository for automation data such as policies.
- A first level automation domain. This is actually the automation target. It is mainly a single node or a group of nodes running the same operating system. The interface to the end-to-end automation manager is established through an automation adapter.

  The automation management component consists of a solution and operations console that interfaces with the browser and a operations console that establishes the interface to the end-to-end automation manager.

High availability and resource monitoring

Tivoli System Automation provides a high availability environment. High availability, as mentioned, describes a system where downtimes are minimized and that has a self-healing infrastructure to prevent downtime caused by system problems. Such an infrastructure detects improper operation of systems, transactions, and processes, and tries to perform corrective action without disrupting users.

Tivoli System Automation offers mainframe-like high availability by using fast detection of outages and sophisticated knowledge about application components and their relationships. It provides consistent recovery of failed resources and whole applications on systems in a clustered environment without operator intervention.

Policy based automation

Tivoli System Automation allows the configuration of high availability systems based on policies. The policies define the relationship between components of an application. These policies can be applied to existing applications with minor modifications.

Once the relationships are established, Tivoli System Automation will assume responsibility for managing the applications on the specified nodes as configured. In other words, Tivoli System Automation exclusively starts, stops, and monitors the components.

Automatic recovery

Tivoli System Automation quickly and consistently performs an automatic restart of failed resources or whole applications either in place or on another system of a cluster. This greatly reduces system outages.

Automatic movement of applications

Tivoli System Automation manages the cluster-wide relationships among resources for which it is responsible. If applications need to be moved among nodes, the start and stop relationships, node requirements, and any preliminary or follow-up actions are automatically handled by Tivoli System Automation. This again relieves the operator from manual command entry, reducing operator errors.

Resource grouping

Resources can be grouped together in Tivoli System Automation. Once grouped, all relationships among the members of the group can be established, such as location relationships, start and stop relationships, and so on. After all of the configuration is completed, operations can be performed against the entire group as a single entity. This once again eliminates the need for operators to remember the application components and relationships, reducing the possibility of errors.

4.4.3 SteelEye LifeKeeper

SteelEye is a data and application availability management solution for business continuity and disaster recovery on Linux and Windows. For information about SteelEye please refer to the following Web page:

http://www.steeleye.com/

The SteelEye LifeKeeper family of application-focused data replication, high availability clustering, and disaster recovery solutions are easy to deploy and operate, and enable enterprises of all sizes to ensure continuous availability of business-critical applications, servers, and data. The solutions are proven in the most demanding of environments and are integrated to deliver flexibility, scalability, and a fast return on investment.

SteelEye LifeKeeper offers enterprise-grade reliability while simplifying implementation with certified solutions for a wide range of applications and databases running on Windows and Linux, including mySAP™, Exchange, Apache, Oracle, DB2, SQL Server, MySQL, PostgreSQL, and others.

To complement its software solutions, SteelEye also provides a full range of high availability consulting and professional services to assist organizations with the assessment, design, and implementation of solutions for ensuring high availability within their environments.

5

Monitoring and measuring key metrics

The successful deployment of a mission critical application is the first major step in a successful implementation. Running the application is another step of equal importance. During the deployment of the application, some agreements have been negotiated between the service provider and the users of that application. These agreements very likely have found their way into service level agreements.

In this chapter, we will discuss how to monitor critical application behavior and the importance and the role of monitoring efforts in mission critical application environments.

In 5.1, “Introduction” on page 84, we discuss the basic terms monitor and metric and give an overview of application monitoring.

In 5.2, “Monitoring and metrics” on page 85, we discuss the more general implications and develop some strategies of what to monitor and how to monitor.

In 5.3, “Monitoring examples” on page 90, we provide a tour of Linux on POWER and its monitoring capabilities.

5.1 Introduction

In IT environments, a *metric* is the measurement of a particular characteristic of a component's performance or efficiency. A metric is sometimes used directly and sometimes as an element in an algorithm. For example, in programming, a benchmark includes metrics.

During the planning for an application's deployment, some key measurements for the operation of the application have been agreed upon. These key measurements also were eventually normalized into a general description, as shown in Table 5-1.

Table 5-1 Characteristics of a metric

Element	Description
Metric	Name of the metric.
Metric description	Description of what is measured.
Measurement procedure	How is it measured?
Frequency of the measurement	How often is the measurement taken?
Threshold	How are the thresholds calculated?
Target value	What is the best value of the metric?
Unit	Units of measurement.

Once these key measurements or metrics have been defined, they must be monitored. Have a look at Figure 5-1 on page 85. It shows a typical, contemporary application environment, showing from left to right:

- Clients that have access to a network, in most cases to an internet type network.
- To communicate with the application, a front-end HTTP Server accepts connections from clients.
- Transaction requests issued by clients are serviced by an application server. This can be a single instance of the application or multiple instances running on a number of servers or in a number of LPARs.
- The application server normally accesses further information by talking to one or more databases and other back-end services.

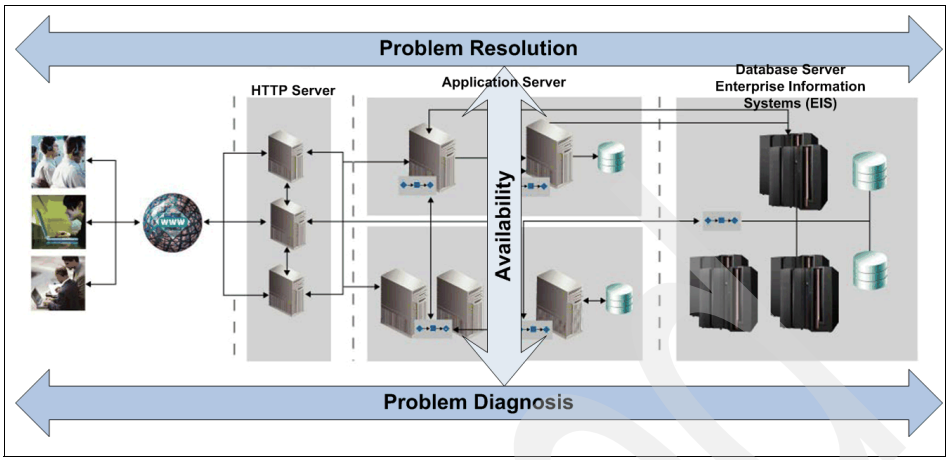


Figure 5-1 Typical application environment

In environments illustrated in Figure 5-1, monitoring is essential to:

- Keep the application available according to the service level agreements in place
- Provide basic data for trend analysis
- Provide data as input for diagnosis and troubleshooting efforts.

5.2 Monitoring and metrics

The importance of a certain metric and the implementation of a monitor depends on the viewpoint.

From a user's viewpoint, any possible performance degradation that might be visible should be monitored and measured.

From a system viewpoint, it is a good idea to monitor the underlying infrastructure and collect information about the performance of the various components.

Finally, the application itself might need some monitoring. There might be some situations where the executed code uses a lot of resources. Monitoring these situations can help find performance problems.

5.2.1 What to monitor

To collect the necessary data, we need to define the resources to be monitored and why they should be included in the monitoring scheme. Generally, required monitors fall into one of the following categories:

- Can the resource being monitored cause an outage?
- Can the resource being monitored cause a degradation of the system in terms of usability or integrity of the system itself?
- In case of failure, can the resource being monitored have an impact on the performance of the system?

Figure 5-2 on page 87 shows our example application environment from another viewpoint. Three distinctive nodes provide the application. On each node, there are resources to be monitored. An analysis of all of them will give you an overall picture about the status and health of your application:

- First of all, it is always a good idea to monitor the essential system resources on a node, no matter if it is a physical server or a LPAR, that is, the CPU I/O and paging behavior should be monitored on each node, whether be it a dedicated server or an LPAR.
- An interesting and important value on an Web server acting as a front end is the number of running Web server threads. Most Web servers limit the number of open HTTP sessions. Exceeding the defined number of HTTP sessions may lead to connectivity failures from a user's viewpoint.
- On modern application servers, the number of open HTTP sessions as well as the overall number of transactions is especially important in environments using TCP/IP for communications.

  Note: TCP/IP limits the number of transactions between two pairs of hosts to about 285 transactions per second. The limitation is caused by the maximum number of TCP ports available (~65412) and the time a connection is kept alive (240 seconds).

- Finally, when a database or any other information system is serving the application, the number of open data connections is a good indicator and can be used to provide both real time and trend informations.

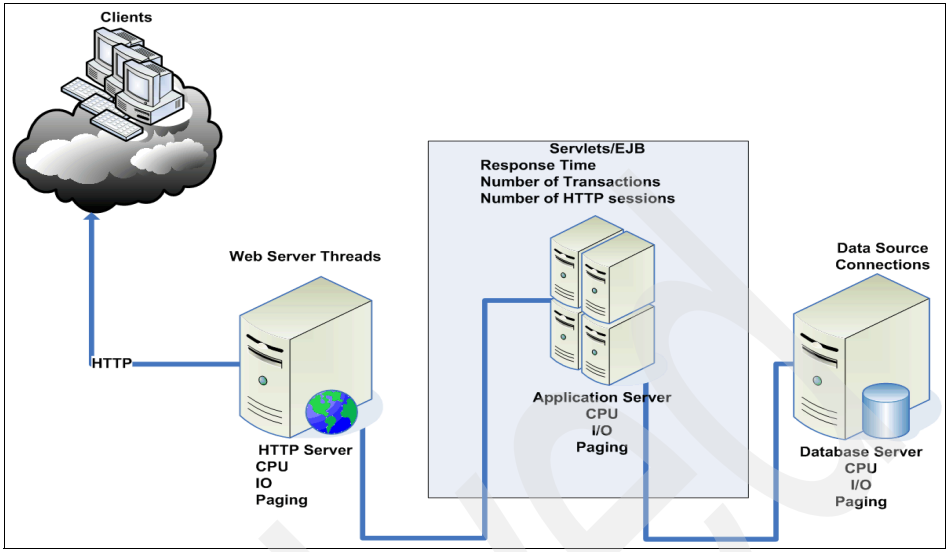


Figure 5-2 Detailed application chain

5.2.2 Monitor or metric types

Finally, before we discuss more monitoring and metric issues, we should look at the types of monitors normally required to get information about the overall state of an application. Keys to the successful deployment and acceptance of an application are:

- The availability of the application from a user's viewpoint
- The security of the environment
- The performance of the application

To monitor the availability of a component or a service, you will use simple up/down monitors. The information about a given resource is that it is up or down, or more generally, the resource is available or not, should be sufficient.

Security and performance metrics and their associated monitors are more difficult to implement. In the case of security, we are normally interested in detecting and preventing attacks against the system. For performance metrics, things like file system and CPU utilization or response time and queued transactions are important. These resources are normally being sampled and then compared against a threshold.

However, there are more components we need to include in the overall picture. Figure 5-3 shows a simplified representation of our example system.

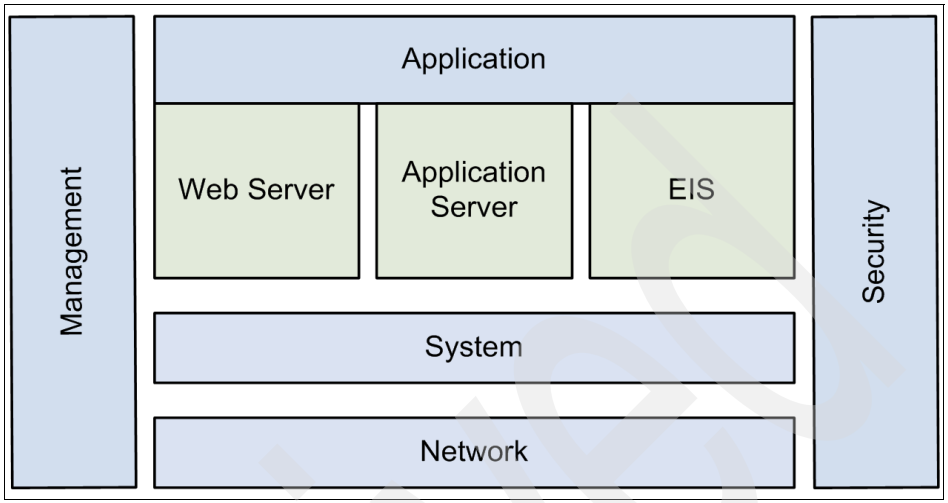


Figure 5-3 An example system layout

Figure 5-3 shows our example application environment with the supporting components, such as network and security components, added. With this layout as a base, we can start to first define the type of metrics we need.

- On the application layer, we need:
  - Response time measurements.
  - Log file analysis (if the application writes a log).
  - Monitoring possible exceptions.
  - On the Web server, we need:
    - An analysis of the log file.
    - Web server statistics.
    - Number of connections.
  - On the application server, if the application is based on a Java virtual machine, the jvm statistics will provide information about the general status. If the application is of another type, such as SAP, it very likely provides its own interface for statistics.
  - On the Enterprise Information Server, there are a number of options that will give the relevant information.

- If you run a database directly at this position, monitoring the table space and buffer pool utilization will give you information about the database health.
- If you run an interface to the back-end mainframe system in that position, monitor the queues.

- On the system layer. The system provides the application with basic resources. Thus, we will monitor these basic resources, namely:
  - CPU utilization.
  - Memory utilization.
  - Disk utilization.
- On the network layer. At the very least, bandwidth, communication speed, and link quality is important. Therefore:
  - The network interface for statistic data provides quality information.
  - Controlled ping commands can provide information about the overall performance.
  - Issuing **`traceroute`** can give us information about possible path problems.

Figure 5-4 summarizes the monitoring options. It shows our sample system layout, including the metrics we just discussed.

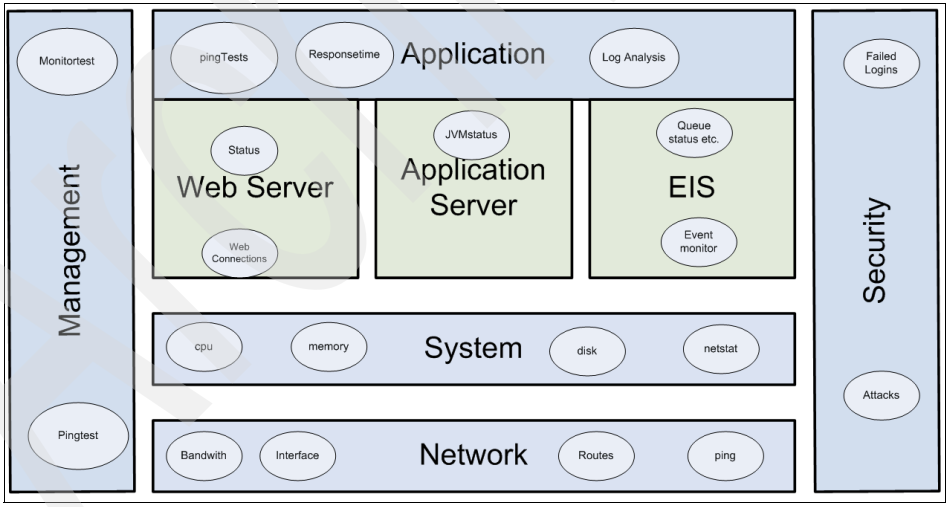


Figure 5-4 The example system layout - metrics added

5.2.3 End-to-end monitoring

Before you implement a monitoring scheme for a given application environment, you should develop a monitoring solution and decide what monitoring tools you would use. Tools can range from ready to use, off the shelf products, such as Tivoli monitoring, over application provided interfaces to operating system commands and developed scripts. In most cases, you will have a mix of these monitoring tools.

Independent from the solution of choice, there are a number of factors that influence the solution:

- What overhead is applied to the whole system by the monitoring component? Obviously, the ideal value would be no overhead. In practice, you should choose the sampling intervals carefully and try to hold the impact of sampling operations to a minimum.
- How scalable is your monitoring solution? If the environment grows, is your solution capable of scaling with the application?
- How complex is the monitoring solution? Does it require extensive maintenance?
- How reliable is your monitoring solution? Does it work with minimum manual intervention once setup? Does it provide the wanted information?

In general, larger enterprises try to implement an end-to-end monitoring solution, which then easily interfaces into other system management disciplines. In practice, it turns out that no single tool exists that sufficiently serves all the monitoring needs. There are always some holes that need to be filled using system provided tools or written scripts.

5.3 Monitoring examples

This section discusses a number of monitoring examples. We focus on Linux on POWER, but in general, these monitors apply to multiple platforms. However, the actual commands may differ, as may the availability of some of the off the shelf tools we mention.

Again, we use our reference application system and talk about monitoring solutions and commands that are generally available for an Linux for POWER system.

5.3.1 Monitoring the system

The key resources to be monitored on a Linux system are CPU, memory, disk I/O, and network interfaces.

Figure 5-5 shows the system areas that are the most common targets for monitoring.

- To monitor CPU performance, Linux provides the **vmstat** and the **sar** utilities.

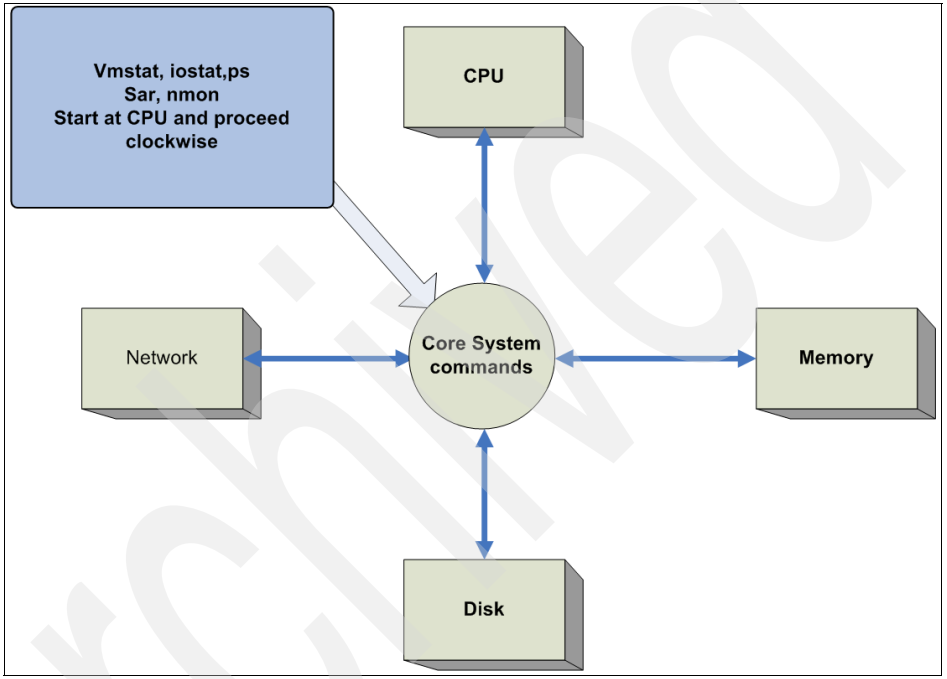


Figure 5-5 System monitoring commands overview

- The next key resource is the available memory. Memory can be monitored using **vmstat**. It gives a very good overview of the actual memory assignments
- For disk monitoring, at least two essential commands can be used. **iostat** provides you with **vmstat** like commands that provide snapshots about the I/O activity. The other command is **df**, which is the tool of choice when you need information about the actual disk space available.
- For the network, we suggest using the **netstat** command, which delivers much more information than just the routes. The **netstat -s** command in particular provides you with a summary of your TCP/IP performance.

A

High availability terminology

This appendix provides some information about the sometimes special terminology used in the high availability solutions introduced in Chapter 4, “High availability solutions” on page 55.

Linux-HA

The following terms were introduced with Linux-HA Version 2 (sometimes referred as Release 2):

- Primitive resources
- Resource Groups
- Resource Clones
- Multi-state Resources

Tivoli System Automation for Multiplatforms terminology

The following terms are used in Tivoli System Automation documentation when describing Tivoli System Automation:

Cluster/peer domain The group of host systems upon which Tivoli System Automation manages resources is known as a cluster. A cluster can consist of one or more systems or nodes. The term “peer domain” is also used when referring to a cluster. The two terms are interchangeable.

Node A single host system that is part of a Tivoli System Automation cluster. Tivoli System Automation V1.2 supports up to 32 nodes within a cluster.

Resource A resource is any piece of hardware or software that can be defined to Tivoli System Automation. Resources have characteristics, or attributes, that can be defined. For example, when considering an IP address as a resource, attributes would include the IP address itself and the net mask.

Resource attributes A resource attribute describes some characteristics of a resource. There are two types of resource attributes: persistent attributes and dynamic attributes.

Persistent attributes: The attributes of the IP address just mentioned (the IP address itself and the net mask) are examples of persistent attributes; they describe enduring characteristics of a resource. While you could change the IP address and net mask, these characteristics are, in general, stable and unchanging.

Dynamic attributes: On the other hand, dynamic attributes represent changing characteristics of the resource.

Dynamic attributes of an IP address, for example, would identify such things as its operational state.

Resource class A resource class is a collection of resources of the same type.

Resource group Resource groups are logical containers for a collection of resources. This container allows you to control multiple resources as a single logical entity. Resource groups are the primary mechanism for operations within Tivoli System Automation.

Managed resource A managed resource is a resource that has been defined to Tivoli System Automation. To accomplish this task, the resource is added to a resource group, at which time it becomes manageable through Tivoli System Automation.

Nominal state The nominal state of a resource group indicates to Tivoli System Automation whether the resources with the group should be Online or Offline at this point in time. So setting the nominal state to “Offline” indicates that you wish for Tivoli System Automation to stop the resources in the group, and setting the nominal state to “Online” is an indication that you wish to start the resources in the resource group. You can change the value of the NominalState resource group attribute, but you cannot set the nominal state of a resource directly.

Equivalency An equivalency is a collection of resources that provides the same functionality. For example, equivalencies are used for selecting network adapters that should host an IP address. If one network adapter goes offline, IBM Tivoli System Automation selects another network adapter to host the IP address.

Relationships Tivoli System Automation allows the definition of relationships between resources in a cluster. There are two different relationship types:

Start-/stop relationships are used to define start and stop dependencies between resources. You can use the StartAfter, StopAfter, DependsOn, DependsOnAny, and ForcedDownBy relationships to achieve this. For example, a resource must only be started after another resource was started. You can define this by using the policy element StartAfter relationship.

Location relationships are applied when resources must, or should if possible, be started on the same or a different node in the cluster. Tivoli System Automation provides the following location relationships: Collocation, AntiCollocation, Affinity, AntiAffinity, and IsStartable.

Quorum

The main goal of quorum operations is to keep data consistent and to protect critical resources. Quorum can be seen as the number of nodes in a cluster that are required to modify the cluster definition or perform certain cluster operations. There are two types of quorum:

Configuration quorum: This quorum determines when configuration changes in the cluster will be accepted. Operations affecting the configuration of the cluster or resources are only allowed when the absolute majority of nodes is online.

Operational quorum: This quorum is used to decide whether resources can be safely activated without creating conflicts with other resources. In the case of a cluster splitting, resources can only be started in the subcluster that has a majority of nodes or has obtained a tie breaker.

Tie breaker

In the case of a tie in which a cluster has been partitioned into two subclusters with an equal number of nodes, the tie breaker is used to determine which subcluster will have an operational quorum.

B

IBM System p RAS characteristics

This appendix lists the software and hardware RAS features for various System p servers. Table B-1 lists the hardware RAS features for various System p servers.

Table B-1 Summary of RAS features for different System p models

RAS feature	IBM eServer pSeries 505/510	IBM eServer pSeries 520	IBM eServer pSeries 550/550Q	IBM eServer pSeries 570	IBM eServer pSeries 590/595
Service processor	X	X	X	X	X
Redundant service processor	N/A	N/A	N/A	Optional	X
Redundant clock	N/A	N/A	N/A	N/A	X
System deallocation of failing components	X	X	X	X	X
Memory and L3 cache availability functions	X	X	X	X	X
Memory spare from CUoD at IPL	N/A	N/A	N/A	X	X
Processor and L1/L2 cache RAS functions	X	X	X	X	X
Dynamic processor spare from CUoD	N/A	N/A	X	X	X
IO Drawer 7311-D10 and 7311-D11	N/A	N/A	N/A	X	N/A
IO Drawer 7311-D20	N/A	X	X	X	N/A

RAS feature	IBM eServer pSeries 505/510	IBM eServer pSeries 520	IBM eServer pSeries 550/550Q	IBM eServer pSeries 570	IBM eServer pSeries 590/595
IO Drawer 7040-61D (PCI-X only)	N/A	N/A	N/A	N/A	X
I/O Drawer redundant connections	N/A	X	X	X	X
Redundant or N+1 power supply and fan	Optional	Optional	Optional	X	X
Integrated Battery Backup	N/A	N/A	N/A	N/A	Optional

Table B-2 lists the software RAS features for the various operating systems that run on System p servers.

Table B-2 Operating system support for selected RAS features

RAS feature	AIX 5L™ V5.2	AIX 5L V5.3	i5/OS	RHEL AS V3	RHEL AS V4	SLES V9
System deallocation of failing components						
Dynamic processor deallocation	X	X	X	-	X	X
Dynamic processor sparing at runtime	X	X	X	X	X	X
- Using CUoD processors	X	X	X	X	X	X
- Using capacity from an active spare pool	X	X	X	X	X	X
Persistent processor deallocation	X	X	X	X	X	X
Manual memory sparing from active pool	X	X	X			
Memory sparing with CUoD at IPL time	X	X	X	X	X	X
GX+ bus persistent deallocation	X	X	X	-	-	-
PCI bus extended error detection	X	X	X	X	X	X
PCI bus extended error recovery	X	X	X	-	-	Limited
PCI-PCI bridge extended error handling	X	X	X	-	-	-
Redundant RIO link	X	X	X	X	X	X
Hot-swap disks (internal/external)	X	X	X	X	X	X
PCI card hot-swap	X	X	X	-	-	X
Service processor failover at IPL time	X	X	X	X	X	X
System clock failover at IPL time	X	X	X	X	X	X
Memory availability						
ECC memory, L2, and L3 cache	X	X	X	X	X	X
L1 parity check plus retry	X	X	X	X	X	X
Dynamic bit-steering (spare memory in main store)	X	X	X	X	X	X
Chipkill memory	X	X	X	X	X	X
Memory scrubbing	X	X	X	X	X	X
Improved L3 cache line delete	X	X	X	X	X	X
Array recovery L1, L2, and L3 cache	X	X	X	X	X	X
Special uncorrectable error handling	X	X	X			
Fault detection and isolation						
System FFDC diagnostics	X	X	X	X	X	X
IO subsystem FFDC diagnostics	X	X	X	-	-	X

RAS feature	AIX 5L™ V5.2	AIX 5L V5.3	i5/OS	RHEL AS V3	RHEL AS V4	SLES V9
Runtime diagnostics	X	X	X	Limited	Limited	Limited
Error log analysis	X	X	X	X	X	X
Service processor support for:	X	X	X	X	X	X
- Built-In Self Tests (BIST) for logic and arrays	X	X	X	X	X	X
- Wire tests	X	X	X	X	X	X
- Component initialization	X	X	X	X	X	X
Serviceability						
Boot-time progress indicator	X	X	X	Limited	Limited	Limited
Firmware error codes	X	X	X	X	X	X
Operating system error codes	X	X	X	Limited	Limited	Limited
Inventory collection	X	X	X	X	X	X
Environmental and power warnings	X	X	X	X	X	X
Hot-plug fans, power supplies, and power regulators	X	X	X	X	X	X
Extended error data collection	X	X	X	X	X	X
Service processor “call home” with no HMC	X	X	X	X	X	X
RIO cable unplug at runtime	X	X	X	X	X	X
Service processor mutual surveillance with Power Hypervisor	X	X	X	X	X	X
Dynamic firmware update with HMC						
Service Agent	X	X	X	-	-	X
System dump for memory, service processor, and Hypervisor	X	X	X	X	X	X
Operating system error reporting to HMC SFP application	X	X	X	X	X	X
RMC secure error transmission subsystem	X	X	X	X	X	X
Health check schedule operations with HMC	X	X	-	X	X	X
Automated server recovery/start	X	X	X	X	X	X
Clustering						
High availability clustering with HACMP™ with direct IO	X	X	-	-	-	-
High availability clustering with HACMP and VIOS		X	-	-	-	-

Related publications

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

IBM Redbooks publications

For information about ordering these publications, see “How to get IBM Redbooks publications” on page 103. Note that some of the documents referenced here may be available in softcopy only.

- *End-to-end Automation with IBM Tivoli System Automation for Multiplatforms*, SG24-7117

Other publications

These publications are also relevant as further information sources:

- Tanenbaum, *Operating Systems Design & Implementation*, Prentice Hall, 1987, ISBN 9780131429383

Online resources

These Web sites and URLs are also relevant as further information sources:

- Additional information about Linux on POWER

  http://www.ibm.com/developerworks/linux/power/

- IBM DB2 Database

  http://www.ibm.com/software/data/db2/udb/edition-ese.html

- IBM Server Storage

  http://www.ibm.com/servers/storage/

- IBM Storage SAN information

  http://www.ibm.com/servers/storage/san/

- IBM System p Web site

  http://www.ibm.com/systems/p/hardware/

- IBM Tape libraries

  http://www.ibm.com/servers/storage/tape/

- IBM Tivoli Software

  http://www.ibm.com/software/tivoli/

- IBM Tivoli System Automation

  http://www.ibm.com/software/tivoli/products/sys-auto-linux/

- Linux at IBM

  http://www.ibm.com/developerworks/linux/

- Linux-HA Web site

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

- Linux on POWER Wikis

  http://oss.gonicus.de/openpower/index.php/Main_Page
  http://www.ibm.com/collaboration/wiki/display/LinuxP/Home

- Netcraft

  http://www.netcraft.com

- Novell SUSE Linux

  http://www.novell.com/products/server/overview.html
  http://www.novell.com/products/linuxenterpriseserver/

- Oracle Database

  http://www.oracle.com/index.html

- Product offerings with Linux on POWER

  http://www.ibm.com/systems/p/linux/

- Red Hat Enterprise Linux Web sites

  http://www.redhat.com/software/rhel
  http://www.redhat.com/rhel

- Red Hat Web site

  http://www.redhat.com

- SAP Sizing

  http://service.sap.com/sizing
  http://service.sap.com/quicksizing

- SteelEye LifeKeeper

  http://www.steeleye.com/

- Tape storage compatibility list

  http://www.ibm.com/servers/storage/media/index.html

How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks publications, IBM Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy IBM Redbooks publications or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

T

U

V

W

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Deploying Mission Critical Applications with Linux on POWER

Redbooks

Mission critical solution methodology

Highly reliable IBM System p hardware

Highly available Linux deployments

This IBM Redbooks publication gives you the information you need to help you run mission critical applications in an IBM System p environment. Matching applications, such as SAP, and Linux highly available solutions with IBM System p systems provides you with the capabilities to produce a highly available solution.

This book will help you think through the process of setting up new solutions or help you review an existing solution. If you are on an previous generation IBM POWER server or running Linux on another architecture, this book will help you understand the benefits of moving your critical applications to Linux running on IBM System p servers.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

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

For more information:
ibm.com/redbooks

SG24-7286-00 ISBN 0738466604