

Implementation of a Disaster Resilient Linux Cluster with Storage Subsystem Based Data Replication

WERNER FISCHER

DIPLOMARBEIT

eingereicht am
Fachhochschul-Diplomstudiengang
COMPUTER- UND MEDIENSICHERHEIT
in Hagenberg

im Juni 2004

© Copyright 2004 Werner Fischer

Alle Rechte vorbehalten

Erklärung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und die aus anderen Quellen entnommenen Stellen als solche gekennzeichnet habe.

Hagenberg, am 4. Juni 2004

Werner Fischer

Contents

Erklärung	iii
Preface	viii
Kurzfassung	ix
Abstract	x
1 Introduction	1
1.1 What is Linux?	2
1.2 Why Linux?	2
1.3 Why is it a worth-while question?	2
1.4 Overview of Main Results	3
1.5 Structure of the Thesis	3
2 Review of HA Cluster Technologies	5
2.1 Shared-Everything Cluster	5
2.2 Shared-Nothing Cluster	5
2.3 Shared Resource Protection	6
2.3.1 Lock Managers	6
2.3.2 Quorum Mechanisms	7
2.3.3 Fencing Mechanisms	8
3 Disaster Recovery Basics	10
3.1 Tier Levels of Disaster Recovery	10
3.2 Networking Requirements	11
3.2.1 Rapid Spanning Tree Protocol	11
3.2.2 Ethernet Channel Bonding	12
3.3 Possible Configurations	12
3.3.1 Two Sites with Equal Number of Nodes	13
3.3.2 Two Sites with a Dominant Site	13
3.3.3 Three Sites	15

4	Data Replication	16
4.1	Requirements for Data Replication	16
4.2	Symmetric and Asymmetric Data Replication	16
4.3	Data Replication Mechanisms	17
4.3.1	Synchronous Replication	17
4.3.2	Non-Synchronous Replication	17
4.3.3	Asynchronous Replication	18
4.3.4	Combination of Synchronous and A- or Non-synchronous Replication	19
4.4	Data Replication Levels	19
4.4.1	Data Replication at Server Level	19
4.4.2	Data Replication at Storage Subsystem Level	19
4.4.3	Data Replication within the SAN	20
4.5	Examples for Storage Subsystem Based Data Replication	20
4.5.1	IBM FAStT Remote Volume Mirroring	20
4.5.2	IBM ESS Peer to Peer Remote Copy	20
4.6	Allow Data Writes if Mirroring is Impossible?	21
4.6.1	Allow Data Writes without Mirroring	21
4.6.2	Deny Data Writes without Mirroring	21
5	State of the Art in Disaster Recovery	24
5.1	IBM GDPS for zSeries	24
5.1.1	GDPS/PPRC	24
5.1.2	GDPS/XRC	25
5.2	IBM eRCMF	25
5.3	IBM HACMP/XD	25
5.4	HP-UX Disaster Tolerance Clusters	26
5.4.1	HP Extended Distance Clusters	26
5.4.2	HP Metropolitan Cluster	26
5.4.3	HP Continental Cluster	27
5.5	Sun Cluster 3.x	27
5.6	OpenVMS Cluster	28
6	Problem Statement	29
6.1	What is it all about?	29
6.2	Analysis of today's Solutions	30
6.2.1	No Linux Support	30
6.2.2	Source Storage Devices at one Site only	30
6.2.3	Mandatory Quorum Devices at a third Site	30
6.3	What is the Benefit?	30

7	Implementation	31
7.1	Implementation Overview and Requirements	31
7.2	Cluster Manager Requirements	32
7.2.1	Support for more than two nodes	32
7.2.2	No mandatory quorum device	32
7.2.3	No mandatory fencing mechanisms	33
7.3	Storage Automation via Command Line Interface	33
7.4	Features of the Implementation	33
7.5	PPRC	33
7.5.1	PPRC Basics	33
7.5.2	Establishing PPRC Paths	34
7.5.3	Necessary PPRC Tasks	35
7.6	Service-dependent PPRC Configuration Files	37
7.7	The pprcvolume Script	39
7.7.1	start Operation	39
7.7.2	stop Operation	42
7.7.3	PPRCfailback Operation	42
7.7.4	status Operation	43
7.7.5	statusReport Operation	43
7.7.6	establishPPRC Operation	43
7.7.7	reEstablishPPRC Operation	43
7.7.8	terminatePPRC Operation	44
7.7.9	skipPPRCfailback Operation	44
7.8	Logging Configuration	44
8	Tests	46
8.1	Test Environment	46
8.1.1	Tivoli System Automation Configuration	47
8.1.2	pprcvolume Configuration	47
8.2	Initial Setup and Synchronization	47
8.3	Site A down	47
8.4	ESS A down	48
8.5	One Node at Site A down	49
8.6	Network Connection between Site A and Site B down	50
8.7	PPRC between ESS A and ESS B down	51
8.8	Split Brain between Site A and Site B	52
8.9	Network at Site A down	52
8.10	Network completely down	53
8.11	Rolling Disaster Situation 1	53
8.12	Rolling Disaster Situation 2	54
9	Conclusions	56
9.1	Conclusions	56
9.2	Possibilities of Future Enhancements	56

A Configuration Files	58
A.1 pprcvolume Configuration for nfserver	58
A.2 pprcvolume Configuration for mysql	59
B Contents of CD-ROM	61
B.1 Diploma Thesis	61
B.2 <i>LaTeX</i> -Files	61
B.3 Implementation	62
B.4 Test configuration	62
B.5 Bibliography	62
Bibliography	63

Preface

This thesis is part of my studies for Computer and Media Security in Hagenberg, Austria. It was written in co-operation with IBM Deutschland GmbH, Mainz.

I thank Alexander Warmuth, my tutor at IBM, for the great support during my internship and writing of my thesis. Also thanks to Robert Kolmhofer, assessor at the University of Applied Sciences Hagenberg. I thank all the colleagues working in the Advanced Technical Support at IBM Mainz for the input, ideas, and discussion during coffee breaks. Their hints made me think about things again and again. Another big help was the support of the Tivoli System Automation Team from Böblingen. Thank you Enrico and Thomas! Thanks also to Paul McWatt, who did another proof-read of the thesis.

Last, but not least, I want to thank my parents for the big support through the four years of my studies.

Kurzfassung

Hochverfügbare EDV Dienste gewinnen zunehmend an Bedeutung. Der höchste Grad an Verfügbarkeit kann durch Disaster-beständige Cluster, die über zwei oder mehrere Rechenzentren aufgebaut sind, erreicht werden. Implementierungen von derartigen Architekturen gibt es für einige kommerzielle UNIX Betriebssysteme sowie für Mainframes. Lösungen für Linux fehlen nach wie vor in diesem Gebiet.

Ein springender Punkt bei der Implementierung eines Disaster-beständigen Clusters ist die Spiegelung oder Replizierung von Daten. Moderne Speichersysteme bieten eingebaute Hardwarefunktionen für die Datenspiegelung. Diese haben einige Vorteile gegenüber Software-basierter Datenspiegelung. Hardware-basierte Spiegelung führt im Gegensatz zur Software-basierten zu keiner erhöhten Systemlast auf Cluster-Knoten. Ein weiterer Vorteil ist die Transparenz von Hardware-basierter Spiegelung gegenüber Cluster-Knoten. Dies vereinfacht das Hinzufügen, Ersetzen oder Entfernen von Knoten.

In der Diplomarbeit werden einige allgemeine Bereiche von hochverfügbaren Computer-Systemen diskutiert. Zu Beginn wird ein Überblick über Hochverfügbarkeitstechnologien für lokale Cluster gegeben und einige Grundlagen über Disaster Recovery erklärt. Weitere Themen sind Details über Mechanismen zur Datenreplizierung und der derzeitige Stand der Technik im Bereich Disaster Recovery. Die Problemdefinition beschreibt das Fehlen von Linux-Lösungen und die Einschränkungen von vorhandenen Lösungen für andere Betriebssysteme. Als mögliche Lösung wird ein Prototyp für die Automatisierung der Datenreplizierung durch Knoten eines Linux-Clusters präsentiert. Das Verhalten dieses Prototyps wird im Testkapitel erläutert.

Abstract

Highly available IT services are becoming more and more important. The highest level of availability can be achieved with disaster resilient clusters spanned across two or more computer center sites. Implementations of those architectures are available for a number of commercial UNIX operating systems or mainframe systems. There is still a lack of Linux solutions in this area.

One key point of a successful disaster resilient cluster implementation is the mirroring or replication of data. Modern storage subsystems provide built-in hardware functions for this data mirroring. They have some advantages compared to software-based mirroring. Unlike software-based mirroring, they do not add any processing load to cluster nodes. Another benefit is that mirroring is transparent to nodes through the SAN (Storage Area Network). This makes adding, replacing or removing of nodes easily possible.

The thesis discusses some general areas of high availability computing. It starts with a review of high availability technologies for local clusters and some disaster recovery basics. The next topics detail data replication and the current state of the art in disaster recovery solutions. The Problem Statement describes the lack of Linux solutions and limitations of solutions for other operating systems. As a possible solution, a prototype for the automation of data replication through Linux nodes is presented. The behavior of this prototype in disaster situations is shown in the Tests chapter.

Chapter 1

Introduction

Today many businesses need their IT services continuously available. Therefore, High Availability (HA) Clusters are often used to prevent single points of failure in the data center. In the case of a complete site disaster, these cluster solutions cannot help any more as they lack cluster nodes located at a different data center site.

Other solutions can integrate cluster nodes at different sites, they are known as *geographically dispersed clusters*. These products often are normal HA cluster managers with small modifications. Many solutions use a software replication mechanism to keep a current copy of data at the second site (known as disaster recovery site). Software based replication mechanisms have some drawbacks, e.g. they cause an extra load on the hosts and have to be maintained on all nodes of the cluster.

With the use of Storage Area Networks (SANs) in the computing environment, all data is stored on centralized storage subsystems. These subsystems often provide hardware functions for data replication. As these hardware based functions have been designed to work in disaster recovery situations also without application cluster automation, they imply manual interventions to accomplish a fail-over to the recovery site. Fully automated solutions are only available for Mainframe computing environments and some commercial UNIX operating systems.

The goal of the diploma thesis is the integration of automated storage mirroring management functions with common Linux high availability cluster managers. The implementation is demonstrated in conjunction with IBM Tivoli System Automation for Linux (TSA)¹ as cluster manager. TSA is only used as an example. The implementation itself is principally independent of the cluster manager used. With minor adaptations the prototype should also work with other UNIX operating systems.

¹<http://www.ibm.com/software/tivoli/products/sys-auto-linux>

1.1 What is Linux?

The name Linux² refers to the kernel originally developed by Linus Torvalds. Most people use the name Linux also for the GNU/Linux³ operating system. When talking about Linux, the complete GNU/Linux operating system is meant through this thesis.

1.2 Why Linux?

Linux becomes more and more common, also in data center environments. One big advantage is the number of ports of Linux for different hardware architectures. No other UNIX derivate supports more hardware architectures than Linux. With a broad usage of Linux in the company, system administrators can use their Linux knowledge on all the different hardware architectures where Linux is used.

Linux is Open Source. Open Source Software has many advantages compared to commercial software. Without the power of Open Source Software Linux would not be available for so many different hardware architectures.

Disaster resilient cluster solutions with storage subsystem based data replication are partly available for other UNIX operating systems. Linux has been neglected in this area up to now.

As Linux (both the kernel and the whole distribution) is highly customizable, it can be cut down to provide only the functions that are really needed. This saves computing power and minimizes the risks of security flaws (the more software is running on a system, the higher is the probability of security flaws). Both characteristics (economical use of computing power and a high level of security) are significant for high availability clusters.

1.3 Why is it a worth-while question?

In a disaster situation automated processes will help to minimize downtime and to prevent human errors. The white-paper *IBM Storage Infrastructure for Business Continuanace* [KP03] mentions ten lessons learned from 9/11. The most important point is “Successful recovery necessitates a greater dependency upon automation rather than people”. Crucial IT administrators may not be available in a disaster situation to execute critical tasks for a site fail-over. As a result, all necessary steps for a successful fail-over must be as simple as possible. This goal can only be achieved with a high level of automation.

²<http://www.kernel.org>

³<http://www.gnu.org/gnu/linux-and-gnu.html>

1.4 Overview of Main Results

Designing a disaster resilient cluster architecture is always a big challenge, independent of the operating system. Customer demands on those solutions may differ in many ways. This makes a common solution that fits most companies' needs impossible. The thesis discusses the main challenges and gives an idea of how disaster resilient clusters can be implemented with storage subsystem based data replication. The implemented prototype shows the integration of automated storage management with high availability clusters. This prototype is not suited for production use without further investigation and testing. As mentioned above, disaster resilient clusters must be individually designed for every single implementation.

1.5 Structure of the Thesis

Chapter 1: Introduction

This introduction to the thesis.

Chapter 2: Review of HA Cluster Technologies

This chapter explains some basics of high availability clustering. As disaster resilient clusters are often implemented as geographically dispersed clusters, these principles are also valid for them. Differences between shared-everything and shared-nothing clusters are shown, as well as an overview about shared resource protection.

Chapter 3: Disaster Recovery Basics

Some concepts are discussed in this chapter. The *Tier Levels of Disaster Recovery* help to compare different solutions. Networking requirements necessary for disaster resilient configurations are explained. Finally, possible configurations with two or three computer center sites are discussed.

Chapter 4: Data Replication

Data replication is a very important subject of disaster resilience. The requirements for data replication are explained. Also differences between symmetric and asymmetric replication are explained. Replication mechanisms (synchronous, non-synchronous and asynchronous) are compared, replication levels explained and some example implementations mentioned. Finally there is a discussion about the question "*Allow Data Writes if Mirroring is Impossible?*".

Chapter 5: State of the Art in Disaster Recovery

This chapter examines important disaster recovery solutions. The described products include implementations for mainframes, commercial UNIX operating systems and the OpenVMS operating system.

Chapter 6: Problem Statement

Describes the problem and evaluates today's solutions presented in chapter 5. It gives an outlook of the benefits of a Linux based solution.

Chapter 7: Implementation

This chapter focuses on the implementation of a prototype for automation of data replication management. The prototype is implemented as a BASH⁴ script.

Chapter 8: Tests

Different tests show the behavior of the prototype in certain disaster situations.

Chapter 9: Conclusions

Final conclusions and some "lessons learned" can be found in this chapter.

⁴<http://www.gnu.org/software/bash>

Chapter 2

A Brief Review of High Availability Cluster Technologies

2.1 Shared-Everything Cluster

In a shared-everything cluster, every cluster node can access shared resources like hard disks. To coordinate the access of all nodes, a lock manager is necessary. If a node wants to use a shared resource, it queries the lock manager and asks for access. When it does not use the resource any more, it sends a message to the lock manager so that it releases the access to this resource. Lock managers are implemented either as a single lock manager (SLM), redundant lock manager (RLM) or distributed lock manager (DLM).

To allow parallel access to shared disks, a special file system must be used. For Linux OpenAFS¹, OpenGFS², Sistina GFS³ or IBM GPFS⁴ are available. They all use lock managers to coordinate access to the shared disks.

2.2 Shared-Nothing Cluster

In a shared-nothing cluster, nodes only get exclusive access to a shared resource. While one node owns a resource, no other node may access it in any way. A shared-everything cluster for example would allow concurrent read access to disk volumes. This is not possible in shared-nothing clusters. In the case of a fail-over or manual switch-over of services to another node,

¹<http://www.openafs.org>

²<http://opengfs.sourceforge.net>

³http://www.sistina.com/products_gfs.htm

⁴<http://www.ibm.com/servers/eserver/clusters/software/gpfs.html>

the owning node must release the resource to allow access to it by another node.

In case of split-brain situations (where the cluster nodes are still alive, but cannot communicate with each other) there must be a guarantee that no single resource is owned by two nodes at the same time. To prevent this, different quorum and resource fencing mechanisms are possible as shown later.

2.3 Shared Resource Protection

2.3.1 Lock Managers

Lock managers are mandatory for shared-everything clusters. They can also be used for resource protection in shared-nothing clusters.

Single Lock Manager

If all locking information is stored only at a single server this server is referred to as single lock manager (SLM). As locking information is not available at another server, the SLM is a single point of failure. It is not well suited for high availability clusters. For example, OmniLock from Sistina can be configured as a SLM.

Redundant Lock Manager

To improve this situation, the SLM can be configured in a sub-HA-cluster. This would be a simple two node shared-nothing cluster, which runs the lock manager. The locking functions need not be changed and the whole implementation can be the same as with a SLM. In such configurations, the lock manager is called redundant lock manager (RLM). An example again is OmniLock.

Distributed Lock Manager

Other lock protocols use a distributed lock manager (DLM). OpenGFS uses such a DLM, called OpenDLM⁵. As an example the algorithm used in the OpenDLM implementation [Cah03] is described here. Knowledge about a lock is distributed over three areas:

- Directory
- Master
- Requester/Grantee

⁵<http://opendlm.sourceforge.net>

If a node wants to request a lock for a resource, it sends a query to the responsible directory server for that resource. The information, which cluster node is the directory server for a special resource, is determined by a fixed algorithm. Each node in the cluster acts as directory server for a range of resources. The directory server holds the information about whether a resource has ever been requested before and which node is the master server for the resource. If the resource has never been requested before, the inquiring node will be the master server for this resource. In the other case, the inquiring node sends a message to the already defined master server. In the case that a node dies, the lock information of this node will be lost. The other nodes which accessed resources that were locked by that node will re-request the locks. This re-requesting works like requesting a resource that has never been requested before.

2.3.2 Quorum Mechanisms

Quorum mechanisms are only useful for shared-nothing clusters. They are not suited for shared-everything clusters.

Quorum Disk

This mechanism uses a dedicated shared disk for storing quorum information. The disk must be accessible to all cluster nodes. As all nodes need both write and read access to the shared disk, a special data structure that allows concurrent access has to be used. For example, this could be implemented in the following way:

Every node has its own area to write on the shared disk. Read access is allowed to the whole disk. When a node wants to take over a resource, it must check if no other node owns the resource at that time. It sets a lock indicator in its write-area. This prohibits other nodes from acquiring resources. Then the node writes the information that it owns the resource from now on and releases the lock. If a node loses its connection to the quorum disk it must immediately release all shared resources as other nodes will take them over. To make the detection of failed nodes possible, every node must update a heartbeat information in its write area on a regular basis. If the heartbeat information is out of date, the resource locks for that node are not valid any more.

Real Quorum of Servers (at least three Cluster Nodes)

Another possibility is to work with the real quorum count of cluster nodes. Therefore, at least three cluster nodes must be configured. Cluster nodes may only use shared resources if they are part of the majority of cluster nodes. If a single node cannot communicate with the other nodes anymore,

it must immediately release all resources. In case of an even number of nodes, a tie-breaker must decide which sub-cluster should run the applications.

Tie-Breaker Mechanisms

Tie-Breakers can be used with two-node clusters or combined with real quorum when more than two nodes are used. If a node, that is running services in a two node cluster, fails (or half of the servers in a cluster with more than two nodes), the other node cannot be sure if the node really died or if only the communication with the other node failed. In the latter case, the node that seems to be broken may still use shared resources. Therefore surviving nodes must not start resources in a tie situation. Services that are already running on surviving nodes will remain running. Before the other services can be brought up again, the tie situation must be resolved. The following tie-breaker solutions allow this.

Manual Tie-Breaker With a manual tie-breaker configured, an operator must decide which node may bring up shared resources. In case that half of the nodes are not reachable anymore, one of the surviving nodes asks the operator if it may bring up services. The operator enters a command that allows the node to bring up services only after he has verified that no associated shared resource is in use by another node.

SCSI Tie-Breaker Shared storage is mandatory for this mechanism. All cluster nodes need physical access to a shared SCSI disk. In case of a tie situation the nodes try to get a SCSI Reservation of this shared SCSI disk. Only nodes which can communicate with the node that got the SCSI Reservation may use shared resources. All other nodes must immediately release all resources.

Quorum Server as Tie-Breaker Some shared-nothing cluster implementations use so-called quorum servers for shared resource protection in tie situations. Quorum servers work similarly to lock managers. The difference is that they only grant exclusive access to shared resources. Quorum servers will only be queried in case of tie situations.

2.3.3 Fencing Mechanisms

Quorum mechanisms are software-level protocols which cannot assure in every case that shared resources are not used simultaneously. The reason is that quorum cannot guarantee the release of shared resources used by an unreachable node in a set period of time. If a node is stale and has no quorum anymore, it should release shared resources immediately. But there are situations that can delay the release, as described in [Rob01].

Fencing mechanisms try to quickly and reliably detach stale nodes from shared resources. The difference to quorum mechanisms is the fact that fencing mechanisms do the detachment without any cooperation of the unreachable nodes.

Resource Fencing

Resource fencing uses hardware features of shared resources that guarantee the exclusive use of the shared resource. For a shared SCSI disk, the SCSI reserve or SCSI persistent reserve attribute can assure this exclusive access. If the SCSI disk is a virtual LUN in a SAN, access rights can also be defined in the SAN switch or the storage subsystem itself. If the errant node tries to access the shared resource again, the access is denied through such hardware mechanisms.

Resource fencing must be implemented for each type of resource separately. It requires support from the operating system for the implemented fencing type.

System Reset Fencing (STONITH)

STONITH (Shoot The Other Node In The Head) is another approach of fencing. Errant nodes are reset and so forced to reboot. On startup, the node tries to rejoin the cluster. Many problems that cause a node to fail can be repaired through a reboot. If a node died completely, it cannot reboot anymore. It is always guaranteed that the errant node does not access any shared resource. Other nodes can acquire shared resources without risk.

STONITH is often implemented via smart power switches or Intel's IPMI⁶. A detailed description on STONITH can be found in [Rob01].

⁶Intelligent Platform Management Interface, see <http://developer.intel.com/design/servers/ipmi>

Chapter 3

Disaster Recovery Basics

3.1 Tier Levels of Disaster Recovery

Disaster Recovery can be implemented in different ways, depending on the business needs and the available budget. The different ways can be considered as the Seven Tiers of Disaster Recovery, as described in detail in [Wea04, p. 45]. The model was developed by the SHARE user group¹ in 1992. It can be seen as a common model for disaster recovery solutions and makes it easier to compare different implementations.

Tier 0: No off-site data This first level represents “no disaster recovery possibilities at all”. There is no data backup, no backup hardware or any kind of documentation on how to recover from a disaster. In case of a disaster, recovery usually will be impossible.

Tier 1: Data backup with no Hot Site Tier 1 represents “regular data backups, which are stored off-site”. After a disaster, the copies on the backup media are still available. All new data, which was created after the last backup cycle will be lost. Before the data can be restored, new hardware must be installed.

Tier 2: Data Backup with a Hot Site This Tier is similar to Tier 1, as backup media are stored off-site and will not be destroyed during a disaster at the production site. But at Tier 2 the backup is stored at a secondary site with IT-infrastructure and facilities. Like Tier 1, all new data since the last backup cycle will be lost in case of a disaster at the production site. As IT-infrastructure is available at the secondary site, a shorter recovery time is possible.

¹SHARE is a non-profit, voluntary organization whose member organizations are users of IBM information systems. It was founded in 1955. Details about SHARE can be found at www.share.org.

Tier 3: Electronic vaulting These solutions use techniques from Tier 2. Additionally, some data is electronically vaulted. This allows the creation of mission-critical data backups more often.

Tier 4: Point-in-time copies Tiers 1 and 2 deliver the backup data via the PTAM (Pickup Truck Access Method) to the secondary site. Tier 4 also provides point-in-time copies, but not tape based as the lower Tiers. Tier 4 solutions are disk based and create the point-in-time copies via data links. These copies can be built more often, so less data will be lost in case of a disaster.

Tier 5: Transaction integrity These solutions are often database-driven, e.g. two-phase commit solutions. They provide consistency of data between the production and the recovery site. Data loss will be little (often no data loss at all). Tier 5 solutions can only be implemented at application level.

Tier 6: Zero or little data loss Disaster recovery at Tier 6 is realized via application-independent replication mechanisms (e.g. storage sub-system based data replication). Tier 6 solutions are used whenever no (or only a minimum of) data loss is acceptable. They allow a rapid restore of IT services.

Tier 7: Highly automated, business integrated solution As with Tier 6, data copies at Tier 7 are maintained through application-independent replication mechanisms. The difference is that Tier 7 solutions include a high level of automation, which minimizes the risk of operational errors in the case of a disaster. The implementation of a prototype for a Tier 7 solution for Linux hosts is the content of this thesis.

3.2 Networking Requirements

To reduce the number of necessary site fail-overs it is wise to increase the availability of the network infrastructure. The following technologies help to reduce network outages.

3.2.1 Rapid Spanning Tree Protocol

The Rapid Spanning Tree Protocol (RSTP) is a standard of the Institute of Electrical and Electronics Engineers, Inc. [Ins04]. It is the successor of the Spanning Tree Protocol (STP). The main advantage of the new RSTP is the reduced rebuild time compared to the STP.

RSTP supports, preserves and maintains the quality of the media access control in computer networks. It makes it possible to build redundant paths

between network components. This increases the availability of the network in case of single outages. RSTP fulfills the following requirements [Ins04, p. 24]:

1. It eliminates data loops.
2. It provides highly available network communication by automatic re-configuration of the Spanning Tree topology in case of outages of LAN components.
3. During changes of the network topology (e.g. because of outages), the new active topology will stabilize within a short, known bounded interval. This minimizes the time of network outages between any pair of end stations on the network. While typical STP networks need about 30 to 60 seconds to bring the network up again, RSTP can achieve this within a few hundred milliseconds to about five seconds (depending on the vendor implementation).
4. The active topology is predictable and reproducible. It can be selected through RSTP parameters.
5. RSTP is transparent for end stations on the network.
6. Communication between RSTP components needs only a very small fraction of the total available bandwidth.

Because of the faster recovery time, it is wise to prefer RSTP to STP. A possible setup with RSTP is shown in figure 3.1.

3.2.2 Ethernet Channel Bonding

Ethernet channel bonding² for Linux enables network configurations with no single point of failure. Besides trunking and IEEE802.3ad Dynamic Link Aggregation, Ethernet channel bonding supports active-backup configurations for high availability configurations [Dav00]. With active-backup mode, nodes are connected to two switches. Only one of the two links is active at a time. In case of a outage of a switch, network cable or network card, the backup link will be activated. Nodes in figure 3.1 use Ethernet channel bonding.

3.3 Possible Configurations

All of today's available disaster resilient cluster solutions use some kind of quorum or lock manager for shared resource protection. Fencing mechanisms do not seem to be suitable for geographical dispersed configurations. Both quorum mechanisms and lock managers require a majority of votes to switch

²<http://sourceforge.net/projects/bonding>

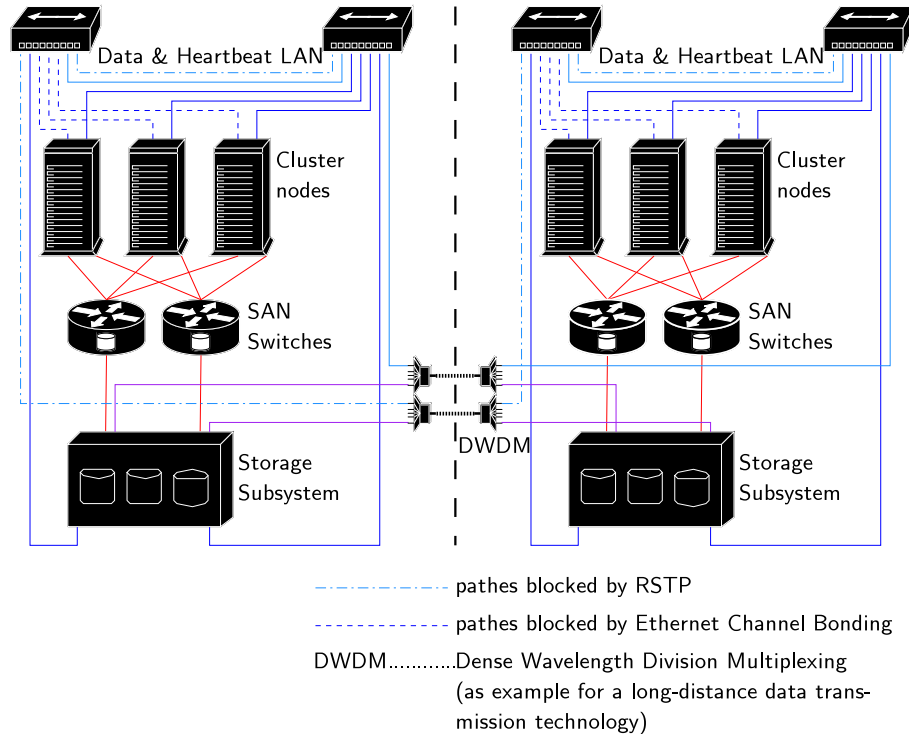


Figure 3.1: Networking in a geo-cluster with RSTP and Ethernet Channel Bonding.

services between nodes. In standard configurations each node has one vote. Some solutions allow an assignment of an user-defined number of votes to each node.

3.3.1 Two Sites with Equal Number of Nodes

At least two different sites are necessary for the implementation of a disaster resilient cluster. The configuration in figure 3.2 shows two sites, each with the same number of votes. A tie situation will occur in case of a split brain or disaster at a site. Manual interaction is necessary to resolve the tie.

3.3.2 Two Sites with a Dominant Site

In this configuration (see figure 3.3) one site owns more votes than the other site. This can be an advantage over the previous solution. If a disaster destroys the site which owns the minor number of votes, the remaining site can automatically take over services.

But there are also drawbacks of this configuration. In case of a disaster at

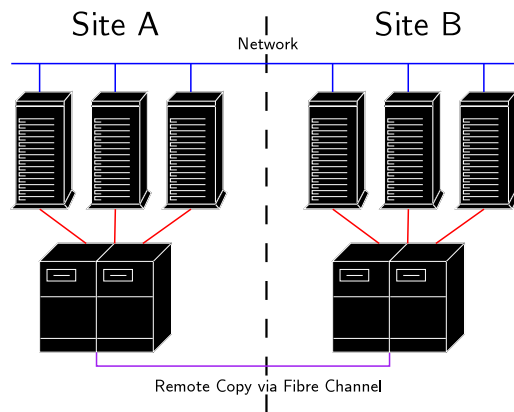


Figure 3.2: Geo Cluster with two balanced sites.

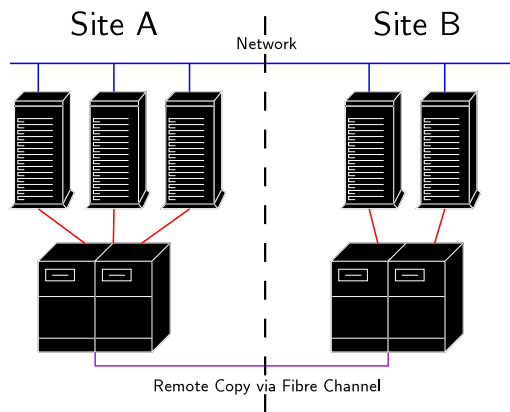


Figure 3.3: Geo Cluster with two sites where one is dominant.

the dominant site, the remaining site alone has less than half of all votes. All running services at the remaining site will be stopped immediately. This is necessary as the remaining site cannot distinguish between a disaster at the dominant site and a split brain situation. In the latter case, the dominant site will bring up all services as it owns more than the half of all votes.

A rolling disaster at the dominant site is even more dangerous. It can happen that a rolling disaster (e.g. a fire) first destroys the communication paths between the two sites. The undamaged site will shut down all services, while nodes at the other site bring up services that were running at the undamaged site. Until the rolling disaster destroys the nodes and the storage system, data which cannot be mirrored anymore will be created. When the dominant site is completely destroyed, all those data updates are lost. This effect is known as “creeping doom scenario” [Fre02].

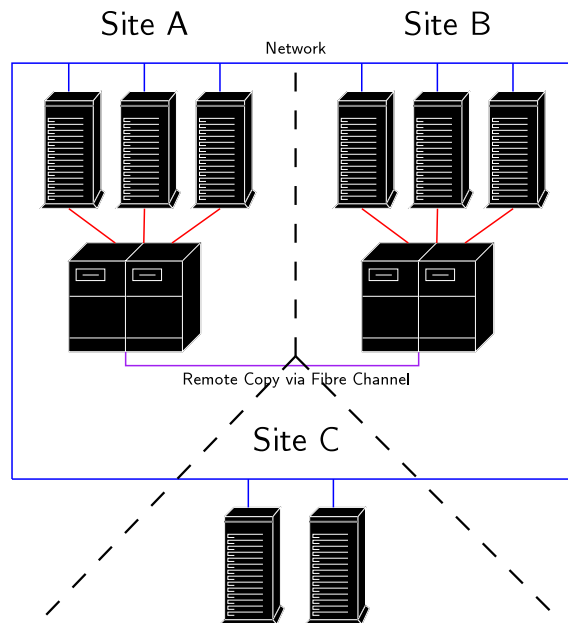


Figure 3.4: Geo Cluster with three sites.

3.3.3 Three Sites

Three site solutions as shown in figure 3.4 allow automatic restart of services after a disaster at a single site. The primary and the secondary site can run services. The third site hosts quorum servers, quorum devices or arbitrator nodes.

In case of quorum servers or quorum devices at the third site, each of the other two sites owns half of all votes. If the two sites cannot communicate with each other anymore, they try to reach the quorum servers or quorum devices at the third site.

Arbitrator nodes at the third site work in a different way. These arbitrator nodes are normal cluster nodes, running the same cluster software as the other nodes. They cannot access the data as no storage system is installed at the third site. So they are configured not to start services in any case. As the arbitrator nodes are normal cluster nodes, the number of votes is distributed across the three sites.

In case of a disaster at site A or site B, all services will be brought up at the remaining site. An outage of site C will never interrupt running services.

Chapter 4

Data Replication

4.1 Requirements for Data Replication

Data replication is a key point in disaster resilient architectures. To prevent data loss, the following areas must be considered:

Data Consistency: preserves the write order of data records. This means that the data is immediately usable and recoverable. Inconsistent data on the other side is not recoverable. Replication mechanisms that produce inconsistent data, like non-synchronous replication, must be combined with other technologies to provide usable point-in-time copies. Data consistency does not mean that the data is current.

Data Currency: indicates how up-to-date data at the recovery site is. The different data replication mechanisms achieve different levels of data currency.

4.2 Symmetric and Asymmetric Data Replication

Most of today's solutions, whether hardware- or software-based, provide asymmetric data replication. Data updates are written from one site to the other, but not in both directions at the same time. Write access is only possible at the primary site, where the source volume is located. The target volume at the secondary site may only be accessible in read-only mode. Before the volume at the secondary site can be mounted, the mirror relationship must be terminated or reversed.

Symmetric data replication can transfer updates from one site to the other in both directions at the same time. Therefore it allows write access to both copies of the data. It requires a locking mechanism to prevent data corruption.

4.3 Data Replication Mechanisms

4.3.1 Synchronous Replication

Synchronous replication mechanisms guarantee the same consistent dataset both at the production and the recovery site as long as the links between the two sites are up and running. This guarantee is achieved as acknowledgments for successful write operations are sent to the calling processes only after the write operation was also successful at the recovery site. Figure 4.1 details the data flow:

1. A process writes data to the storage.
2. The storage subsystem (or some kind of software-based data replication software, like a Logical Volume Manager or Software RAID) sends the write request to a second system at the recovery site.
3. The second system on the recovery site acknowledges the write operation.
4. Now the local storage subsystem (or software-based replication software) sends the acknowledgment also to the calling process.

Although no single data record is lost in case of an outage of the primary storage system, the use of synchronous replication also has limitations. The main problem is the latency caused by waiting for the write acknowledgments. The signal propagation delay between the sites is the main determining factor for this latency. Therefore, the distance between the production and the recovery site is limited. Today's synchronous replication mechanisms support up to 100 km. But those distances can influence the response time of applications noticeably. Extensive testing of the whole system is necessary before going in production use.

There is one case where the logical order of data writes is not guaranteed. During an outage of the replication link the servers on the primary site may continue to write data to the primary storage system (depending on the implementation). If data has been written at the primary site before the replication link comes up again, all new and changed data will have to be transferred to the recovery site. The new and changed data tracks are not written to the recovery site in the same order as they have originally been written to the primary storage system during the time of the outage. During this resynchronization, the data at the recovery site will be inconsistent.

4.3.2 Non-Synchronous Replication

If synchronous replication causes response times at the production site that are too long, non-synchronous or asynchronous replication strategies can help. As the name implies, the data in the two storage systems is not

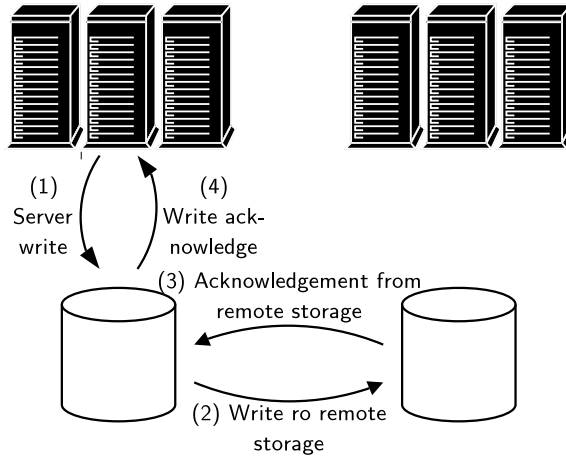


Figure 4.1: Data flow during synchronous replication.

synchronous any more. The storage system at the production site sends an acknowledgment to the calling process as soon as it has written the data locally. After it acknowledges the write, it sends the write operation to the second storage system at the recovery site. With that behavior, the response time at the production site is nearly the same as it would be without replication.

One drawback is that the second system is not completely up to date (data currency). Another big drawback of non-synchronous replication is that write-order is not preserved. Therefore the data on the secondary site is not consistent and cannot be used directly. Consistent snapshots have to be taken at certain periods of time depending on the acceptable amount of last data. To create usable copies of the data at the secondary site, the replication connection is switched to synchronous mode. When synchronization is reached the production applications are stopped and a point-in-time copy is made at the secondary site. After that, the connection can be switched back to non-synchronous and the applications are restarted at the production site.

4.3.3 Asynchronous Replication

Asynchronous replication mechanisms work almost the same way as non-synchronous mechanisms. The main difference is that asynchronous replication preserves the order of writes. Therefore the data set at the secondary system is consistent, although not completely up-to-date.

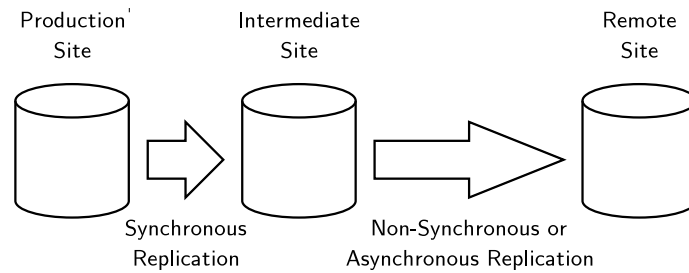


Figure 4.2: Combined synchronous and a/non-synchronous replication.

4.3.4 Combination of Synchronous and A- or Non-synchronous Replication

Whenever synchronous data copies and long distances are needed, replication mechanisms can be combined. For this approach, three sites are necessary. The first site acts as production site, a second site (near enough to enable synchronous replication) as intermediate site, and a third site as remote site. Figure 4.2 illustrates the replication.

4.4 Data Replication Levels

4.4.1 Data Replication at Server Level

Data replication at server level (also known as software-based data replication) implements the storage virtualization within the server itself. Examples are Logical Volumes Managers for commercial UNIX operating systems, Linux Software RAID or the Linux Distributed Replicated Block Device (DRBD) [Rei00]. A detailed overview about different solutions for data replication at server level can be found in [Mit04]. These solutions can be very cost efficient, but they also cause more overhead as soon as more servers participate in the replication domain. Furthermore, as these products operate on the server level, they also cause an extra load on the servers.

4.4.2 Data Replication at Storage Subsystem Level

The main advantage of data replication at storage subsystem level is its invisibility to the application servers. It also causes no extra load on the servers, as they access the LUNs the same way as without data replication. Depending on the data throughput, the performance of the storage subsystem can decrease—an effect that can have an impact on the application.

4.4.3 Data Replication within the SAN

Sometimes, different storage subsystems from different vendors are located at the primary and secondary site. This is often the case after a merge of two companies, when the new firm uses one location as production site and the other as recovery site. Replication approaches at storage subsystem level only work within products from the same manufacturer. With the use of the new virtualization engines, the data replication can be done at the SAN level. This causes no extra load or maintenance on the servers and allows the connection of storage subsystems from different vendors.

4.5 Examples for Storage Subsystem Based Data Replication

4.5.1 IBM FAStT Remote Volume Mirroring

Remote Volume Mirroring (RVM) is a premium feature of the IBM FAStT¹ storage system. It supports synchronous replication only. When a server sends a write request to the FAStT storage subsystem, the data will first be logged in a so-called repository drive. Then the controller writes the data to the primary logical drive, and also sends a remote write operation to the secondary storage subsystem. After the data is stored on both the primary and the secondary logical drive, the log record on the repository drive will be deleted. Now the server gets an I/O completion indication from the primary FAStT.

In case of a link failure, the servers at the primary site can still write to the logical volumes. As the primary FAStT subsystem detects the link failure (or outage of the secondary FAStT) it sends I/O completion indications back to the hosts as soon as the data is written to the local disks. There is no way to change this behavior.

The synchronization algorithm is not capable of partial resynchronizations to start or resume the replication. After every link failure, a full synchronization of all mirrored volumes is performed. This can leave the data at the secondary site in an inconsistent state for a long time.

4.5.2 IBM ESS Peer to Peer Remote Copy

Peer to Peer Remote Copy (PPRC) is the replication technology of IBM's Enterprise Storage Server (ESS)². There are different versions of PPRC available, depending on the ESS model and the version of the installed firmware (licensed internal code—LIC). The following description accounts

¹<http://www.storage.ibm.com/disk/fastt>

²<http://www.storage.ibm.com/disk/ess>

for the latest version of PPRC at the time of writing, PPRCv2 with LIC 2.3.

PPRC supports synchronous (PPRC-sync) and non-synchronous (PPRC-XD) replication. A combination of both is available as Asynchronous Cascading PPRC (see section 4.3.4). For availability and performance reasons, multiple physical links can be used for PPRC connections. The behavior in case of a link failure is configurable. If write operations should be denied in case of a link failure, a feature called Consistency Groups (see section 4.6.2) can be enabled.

The synchronization algorithm is highly optimized. It is possible to suspend a PPRC connection without the need of a full synchronization afterwards. The secondary site can be updated later through the transmission of only the data tracks that are out of sync. This behavior saves bandwidth and also is faster than a full synchronization. During the resynchronization the data of the target volume is not consistent.

The functions “PPRC Failover” and “PPRC Failback” allow the preservation of volume states, even in case of a disaster (as long as the ESS on the production site is not destroyed). After the production site is available again, it is possible to transmit only the changed tracks back to the production site to have both sites synchronous again. Details can be found in [Cea04, p. 417].

4.6 Allow Data Writes if Mirroring is Impossible?

Depending on customer needs, different strategies in case of a mirror link outage are possible.

4.6.1 Allow Data Writes without Mirroring

Some customers need the highest application uptime possible. If all mirroring links (PPRC links) fail, services should continue to run. As a consequence, none of the new data will be mirrored to the other site. Therefore, a rolling disaster which first destroys the PPRC links can be very dangerous. Clients located outside of all computer center sites can still use the highly available applications and issue data writes. For example, these writes could be financial transactions, which cause money transfers to other banks. The debit entries are only written to one storage system. If the rolling disaster destroys this storage system completely later on, the debit entries are lost. But the credit entries may have been transferred to other banks.

4.6.2 Deny Data Writes without Mirroring

Other customers insist on data mirroring whenever applications are running. They would rather stop services for a period of time, instead of allowing

changes to the data. As a consequence applications must be shut down as soon as data mirroring is not possible anymore.

PPRC Consistency Groups

PPRC supports blocking of write operations in case of a complete PPRC link outage through the Consistency Group option [Cea04, p.74]. With this option set, SCSI or FCP (fibre channel protocol) attached open systems receive a QUEUE FULL (QF) status byte code if write operations cannot be completed because of loss of PPRC links. The QF status indicates a full SCSI command queue. Data writes are not placed in the queue. Linux tries to insert the command in the SCSI mid-level queue through the `scsi_mlqueue_insert()` function [You97]:

Generic mid-level SCSI queueing: The point of this is that we need to track when hosts are unable to accept a command because they are busy. In addition, we track devices that cannot accept a command because of a QUEUE_FULL condition. In both of these cases, we enter the command in the queue. At some later point, we attempt to remove commands from the queue and retry them.

The duration of a QF state can be configured in the ESS. The default value for the *Consistency Group Time Out* is set to two minutes.

Reactions on Queue Full

In case of a QF state, applications may behave in different ways. Some may continue to run, others may freeze or cause other unexpected behavior. Depending on the monitoring implementation of the cluster manager, storage or applications will be detected as failed. This causes the cluster manager to fail-over applications. As the applications cannot be started on nodes at the same site, a site fail-over is executed. The volumes at the recovery site will become PPRC sources in suspended state. Applications start at the recovery site, but data writes will not be mirrored.

To prevent this behavior, the cluster manager must be instructed to stop the applications. This is the only secure way to prevent data writes in case of a failure of all PPRC links.

A possible Solution with ESS SNMP Traps

The ESS can be instructed to send SNMP traps if a copy pair in a consistency group becomes suspended and the source volume enters QF. Multiple trap receivers are supported. An enterprise management station should be one of the trap receivers. This ensures notification of system administrators in case of critical situations. Other receivers are `snmptrap-daemons` running on the

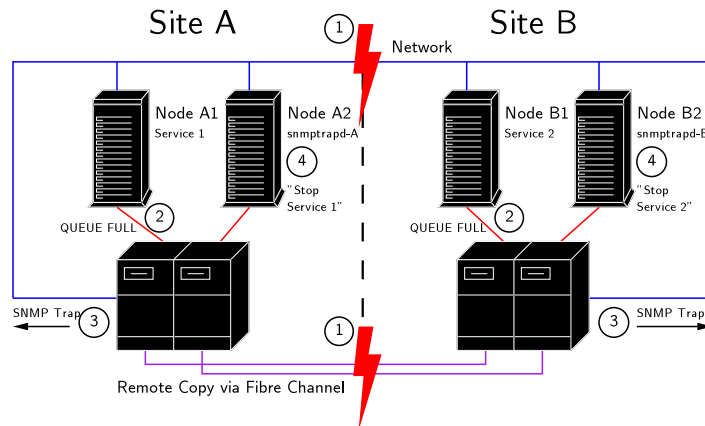


Figure 4.3: Handling of SNMP traps in a split-brain situation.

cluster nodes, one at each site. The two snmptrap-daemons are configured as high availability services within the cluster. The daemon responsible for site A may only run on nodes located at this site. The same applies for the daemon at site B. The two daemons are necessary to guarantee the right behavior even in case of a split-brain situation. It must be possible for each site to separately shut down all services cleanly.

An example for an automated shutdown of applications is shown in figure 4.3:

1. A split brain situation breaks all PPRC and network links.
2. Nodes receive a QF state on write requests.
3. ESS' at both sites send SNMP traps to the trap receivers.
4. Trap receivers execute stop commands for the applications.

Unfortunately SNMP traps do not have guaranteed delivery. If the LAN connection to an ESS is broken, SNMP traps cannot be sent to the trap receivers. It is possible to monitor the sending of SNMP traps through test traps. The sending of those test traps can be caused by cluster nodes through special command line interface commands (`esscli create snmp`). If these test traps cannot be received by the trap receivers, applications must be stopped. This is necessary as broken PPRC links traps would not be received. Although this may help in most situations, it is still theoretically possible that test traps are delivered correctly while other urgent traps are lost. As a consequence, mission critical applications that absolutely must not run if mirroring is impossible, should not be automated with a graphically dispersed cluster.

Chapter 5

State of the Art in Disaster Recovery

The following products and service offerings are examples for Tier 6 and Tier 7 disaster recovery solutions. They are either available for IBM z/OS, IBM AIX or HP UX and are described to give an overview of the possibilities in automated disaster recovery.

5.1 IBM GDPS for zSeries

GDPS (Geographically Dispersed Parallel Sysplex) is a service offering from IBM that implements a Tier 7 solution for IBM's z-Series mainframes¹. There are different versions of GDPS. All of them use hardware-based data replication. Details about GDPS can be found in [Kea04, p. 333] and [Wea04, p. 274].

5.1.1 GDPS/PPRC

GDPS/PPRC uses synchronous PPRC and allows applications to run at both sites at the same time. However, all applications (whether they run at the primary or secondary site) will read from and write to the disk system at the primary site. This can lead to longer response times for applications running at the secondary site. An application running at the secondary site writes data to the primary disk subsystem. Then, the data must be mirrored back to the secondary site. After the primary disk subsystem gets the information that the data has been mirrored, it sends the acknowledgment back to the application.

Logical Units (virtual disks) for open systems (Unix, Microsoft Windows, Linux, Novell Netware and so on) can be integrated into GDPS/PPRC. This

¹<http://www.ibm.com/servers/eserver/zseries>

function—Open LUN management—has been implemented because dependencies between mainframe and open system applications became more and more frequent. GDPS/PPRC only integrates the data replication automation for open systems. It does not automate the restart of open system applications.

5.1.2 GDPS/XRC

GDPS/XRC is an asynchronous disaster recovery solution and is based on Extended Remote Copy (XRC). XRC is a combined hardware and software remote copy implementation. It uses time stamps to preserve the order of write operations. As the data replication between the primary and secondary site is asynchronous, the data at the secondary site will always be slightly behind the data at the primary site. As a benefit XRC provides data replication without any distance limits and only little application latency. As XRC manages the data consistency itself, GDPS must only provide the automation of the recovery process. More details about XRC can be found in [Kea04].

5.2 IBM eRCMF

Enterprise Remote Copy Management Facility (eRCMF) also is a service offering from IBM. It can be implemented as a Tier 4 or Tier 6 solution for open system disaster recovery [Wea04, p. 286]. eRCMF is well suited for very large amounts of data, as it can preserve data consistency across multiple Enterprise Storage Servers. eRCMF uses productivity center machines (PCMs), running on dedicated AIX servers.

eRCMF does not automate the restarting of servers and applications. It makes data volumes directly available at the secondary site after a disaster at the primary site. There is no need to bring up volumes and to start replication fail-over tasks manually. When using synchronous PPRC, eRCMF works as a Tier 6 disaster recovery solution. eRCMF also supports PPRC-XD (non-synchronous data replication) as a Tier 4 solution. The creation of the necessary point-in-time copies can be automated with eRCMF.

5.3 IBM HACMP/XD

HACMP is a high availability clustering solution for local clusters running AIX operating systems. The optional package HACMP/XD also allows fail-over of services to nodes running at a different site. HACMP/XD can be implemented with synchronous PPRC for distances up to 103 km between the sites or with IP based mirroring for unlimited distance support. Detailed information about HACMP/XD can be found in [Wea04, p. 296]

and [Int03a].

HACMP/XD with PPRC automates the PPRC management. There is no need to start fail-over replication tasks manually in case of a disaster. The nodes of the HACMP cluster must have network access to the Enterprise Storage Servers to execute preconfigured tasks when a fail-over or fail-back is necessary.

HACMP/XD with IP based data mirroring can be configured as a synchronous or an asynchronous replication solution. This makes unlimited distance support for asynchronous configurations possible. IP based data mirroring uses Geographic Mirror Devices (GMDs). The same software is used in the Geographic Remote Mirror for AIX (GeoRM), which can be used itself as a Tier 6 disaster recovery tool.

Note that HACMP is only available for the AIX operating systems.

5.4 HP-UX Disaster Tolerance Clusters

All HP-UX disaster recovery cluster products use HP MC/ServiceGuard cluster technology, which is a high availability solution for local clusters. Details about the HP technologies described below can be found in [Hew04], [Hew03], [BK03] and [Fre02]. Note that although the local cluster ServiceGuard product is available for Linux, at the time of this writing the described disaster tolerant solutions are only available for HP-UX.

5.4.1 HP Extended Distance Clusters

Extended Distance Clusters (also known as Extended Campus Clusters, formerly Campus Clusters) are MC/ServiceGuard clusters that are spanned across different data centers separated by a maximum of 100 km. Each site must have an equal number of cluster members. Data is replicated via HP's Mirrordisk/UX, a server-based data replication software. Arbitration can be done by dual lock disks for clusters of up to four nodes. A split-brain will occur in this configuration, if both the heartbeat and the disk link between the two sites fail. In this case both sites will bring up services, compromising data integrity. Clusters with more than four nodes require a quorum server or arbitrator nodes at a third location. Those configurations will not compromise data integrity, even if the primary and secondary site cannot communicate via heartbeat and disk links.

5.4.2 HP Metropolitan Cluster

Metropolitan Clusters use storage subsystem based data replication for data mirroring. Both HP's Continuous Access XP for XP arrays and EMC's SRDF for Symmetrix storage systems can be used. Arbitrator nodes or a

quorum server at a third location are mandatory for Metropolitan Clusters. No kind of lock disks are supported with this cluster.

5.4.3 HP Continental Cluster

A Continental Cluster consists of two distinct MC/ServiceGuard clusters located at different computer center sites. Each of the two clusters maintains its own quorum. As there is no support for automated fail-over, a third site for arbitration is not necessary. The fail-over is initiated by a human operator.

Continental Clusters work with any data replication mechanism. Pre-integrated solutions for HP's Continuous Access XP and EMC's SRDF are available.

5.5 Sun Cluster 3.x

Automated disaster recovery for Sun Solaris can be implemented with the Sun Cluster 3.1² software. This application-fail-over function is fully integrated with the operating system. Sun Cluster supports a cluster file system, which allows shared write access to volumes from multiple hosts. A maximum of three nodes can participate in a campus cluster configuration, but only two nodes can be used for running applications. The third node can only be an arbitrator node. Both two and three-site configurations are possible. Only the three-site setup guarantees system availability after a disaster at one site. In two-site clusters a quorum device is placed within one of the two sites. If that site fails, the other surviving node is useless as it cannot bring up services without manual interaction. Sun itself states that "experience has shown that this technique is very error prone and requires highly skilled personnel to implement correctly" [Str02, p. 15]. Thereby a two site configuration does not seem to be suited for a disaster tolerant solution at all.

Sun Cluster supports both server and storage subsystem based data replication. For storage subsystem based data replication, Hitachi's TrueCopy is used. In case of an outage of a complete site, many manual steps will be necessary if storage-based data replication is used. This can even include actions that must be taken by a Sun service provider [Sun04, p. 69].

Administration of the Sun Cluster does not seem to be easy, as Sun often points out that well-trained, dedicated people must administer the infrastructure, for example in [Str02, p. 4]. This can be very unfavorable as these professionals may not be available in case of a disaster.

As Sun Cluster supports only two nodes that can run services, no high availability can be implemented within a site. Every failure of a single

²<http://www.sun.com/software/cluster>

component leads to a site fail-over.

5.6 OpenVMS Cluster

OpenVMS³ is a shared everything cluster that uses a built-in distributed lock manager. With a maximum of 96 nodes in a cluster, all servers can write simultaneously to the same files of the cluster file system. The disk volumes can be directly attached to all nodes or other nodes can serve the volumes to the rest of the cluster.

Server-based data mirroring is supported through the Volume Shadowing product, a software-based RAID 1 solution. All write operations are synchronous. Multiple disk drives can participate in this mirroring, they are part of a so-called “shadow set”.

A single OpenVMS cluster can be spanned across two geographically separated data centers. The applications may run at both sites at the same time, as all data will be mirrored symmetrically and synchronously across the different data centers.

OpenVMS uses a quorum strategy for arbitration. Therefore every cluster node gets a number of votes. In a split brain situation (in OpenVMS called “partitioned cluster”), no group of cluster nodes would continue to operate if both sites have the same number of votes. A group of cluster nodes needs at least one more vote than half of the sum of all votes in the whole cluster. It is recommended to configure three sites, where the nodes at the third site can act as tie breakers.

³<http://h71000.www7.hp.com>

Chapter 6

Problem Statement

6.1 What is it all about?

Local high availability clustering solutions for Linux have been available for quite some time. However, a fully integrated solution which is capable of handling hardware-replicated storage devices does not exist yet.

In case of a disaster many steps must be performed manually before the IT services can be brought up again at the recovery site. These steps can be very error-prone if the administrator has no in-depth knowledge about the storage subsystem, its replication mechanisms and the behavior and internal operations of the cluster manager. Even the best trained personnel is no guarantee for the right sequence of actions, as a disaster is always a dramatic situation that makes it very difficult to work. Nobody can assure that these people will be available at the recovery site, as they may not have a chance to reach their workplace or have to deal with personal challenges after the disaster. The most extensive solutions are not worth their effort if they rely on experienced administrators who may not be available at the recovery site.

To improve this situation, a higher level of automation must be achieved. All cluster nodes of the geographical dispersed cluster need the ability to configure the storage subsystem for fail-over and fail-back procedures. As the data replication mechanisms of sophisticated storage subsystems are rather complex and support various states of operations, this problem cannot be solved easily.

There are some rules that must be fulfilled in a highly automated environment:

1. Data consistency must not be violated under any conditions.
2. Every single cluster node must be able to execute all necessary fail-over and fail-back operations.
3. In case of inconsistent replication states of the affected volumes, the

cluster manager must recognize this and stop all automated procedures until human intervention.

6.2 Analysis of today's Solutions

6.2.1 No Linux Support

Today there is no automation software for Linux available to handle hardware-based data replication. The solutions presented in chapter 5 are only available for IBM's z-Series mainframes, HP's OpenVMS and some commercial UNIX operation systems like IBM AIX, HP UX and Sun Solaris.

6.2.2 Source Storage Devices at one Site only

IBM's GDPS for z-Series allows data access only at the primary site's storage subsystem. Although applications may also run at the secondary site, this design can have some performance implications, as described in section 5.1.1.

6.2.3 Mandatory Quorum Devices at a third Site

HP's geographical dispersed cluster products, HP Extended Distance Cluster and HP Metropolitan Cluster, require quorum devices at a third site. HP Continental Cluster on the other side has no support for a third site.

6.3 What is the Benefit?

The implemented prototype enables Linux cluster products to integrate storage subsystem based data replication. This allows the design of disaster resilient Linux cluster configurations.

The prototype supports both two- and three-site configurations. Two-site configurations require the same number of nodes at each site. In case of a site disaster or communication loss between the two sites, manual quorum assignment is necessary. A three-site configuration does not need such manual quorum assignments. Although only two sites can be used for running services, cluster nodes at the third site also have votes for quorum. After a complete site disaster, the remaining two sites have enough votes to get quorum. This allows the automatic recovery of services.

Another advantage is the possibility of active-active configurations. Automation of storage replication can be individually configured for each application. Some applications may run at the primary site, while others run at the secondary site. This allows a better utilization of computing resources.

Chapter 7

Implementation

7.1 Implementation Overview and Requirements

Every Linux cluster manager uses some kind of resources to manage highly available services. E.g. a highly available NFS service consists typically of a mount-resource, an IP-address-resource and the NFS-resource (NFS server application) itself, which are started in this order during startup of the NFS service. The resources can be used like init-scripts with at least a start, a stop and a status parameter.

The implemented prototype provides a storage-resource for Linux cluster managers. This resource can be used as first resource during startup of a service before the mount-resource mounts the file systems. If the storage-resource can be started successfully, the data replication states allow mounting of the corresponding file systems. More details about the start process will be shown in section 7.7.1.

The implementation uses IBM's PPRC (Peer to Peer Remote Copy) of the ESS (Enterprise Storage Server). PPRC was chosen because it supports the following sophisticated data replication functions:

- Quick re-synchronization after outages of the replication links.
- Fibre channel links can be used full duplex for concurrent replication in both directions.
- Automation of replication function via command line interfaces.

The storage-resource prototype must fulfill the following requirements:

- It must preserve data consistency.
- If the current data replication states do not allow data access, all necessary actions to reach a state which may allow data access must be executed automatically.

- If the current states do not allow data access and there is no chance of automatic actions reaching a better state, a detailed error log must be provided.

7.2 Cluster Manager Requirements

There are some requirements that a high availability cluster manager for Linux must fulfill before it can be used with this implementation. At the time of this writing, only Tivoli System Automation for Linux fulfilled these requirements. Therefore it was used as example cluster manager for the test scenario used in chapter 8. The following three requirements must be fulfilled.

7.2.1 Support for more than two nodes

For a disaster resilient configuration it is advantageous when at least three nodes participate in the geographically dispersed cluster. Cluster managers with a maximum of two nodes per cluster do not allow local clustering at a site. With one node at each of both sites the maximum node count is reached. In case of an outage of a single node a complete site fail-over will be necessary. Cluster managers that support more than two nodes will be able to do a local fail-over in such a situation. This has the following advantages:

- The mirror relationship between the two sites does not need to be reconfigured. As a result, a local fail-over is faster than a fail-over to the other site.
- Reconnecting clients to the other site takes more time as routing or DNS information must be updated before clients can reconnect.
- With most storage systems, a site fail-over causes the mirror relationship to be suspended. Data writes at the recovery site are not mirrored back to the production site without human interaction.
- A site fail-over causes more automated actions than a local fail-over does. There is a potential higher risk of failures.

7.2.2 No mandatory quorum device

To make both two- and three-site configurations possible, the cluster manager must not depend on a mandatory quorum disk or quorum server. For two-site configurations, a manual assignment of quorum must be possible.

7.2.3 No mandatory fencing mechanisms

Fencing mechanisms cannot be used with disaster resilient clusters. Both resource fencing and system reset fencing need hardware to support the fencing. If the fencing hardware is destroyed during a disaster at a single site, nodes at the remaining site cannot fence the destroyed components.

7.3 Storage Automation via Command Line Interface

Command Line Interfaces (CLIs) allow the automation of storage system functions. Two different CLIs are available for the Enterprise Storage Server: the Storage Management CLI and the Copy Services CLI. Both are written in JAVA and communicate with the ESS via network connections. All communication is encrypted. In-band communication via fibre channel connections is not supported. If the ESS is not reachable via the network, no ESS commands can be executed via the CLIs. The use of the CLIs is described in detail in [Int03b].

7.4 Features of the Implementation

The written prototype automates nearly the complete PPRC management via the two CLIs. Only PPRC tasks must be created manually via the ESS Specialist Web interface. Every application can be configured individually, as shown in section 7.6. This allows site A to act as production site for the first application and as recovery site for the second. Site B on the other side is the production site for the second application and recovery site for the first. The terms “production site” and “recovery site” are used to explain the concepts of the prototype in connection with PPRC terms. The prototype itself does not care about production and recovery sites.

7.5 PPRC

7.5.1 PPRC Basics

PPRC Paths

PPRC uses so-called PPRC paths to replicate data to another ESS. PPRC paths can be established between two Logical Sub-Systems (LSSes). An ESS consists of 16 LSS, which contain RAID-protected data storage on hard disks. Details about LSSes can be found in [Cea02, p. 68]. PPRC paths can use multiple physical fibre channel links. This enhances the possible throughput and eliminates single points of failure in the replication stream.

PPRC Connections and PPRC States

PPRC connections are configured at volume level. To establish a PPRC connection between two volumes, a PPRC path between the corresponding LSSes must exist. PPRC connections can be synchronous or non-synchronous. Every PPRC connection needs two volumes, one acting as *PPRC source* and one acting as *PPRC target*. If a volume is not part of a PPRC connection, it is referred to as *PPRC simplex*.

A PPRC volume can have one of the following states:

- full-copy
- suspended
- copy-pending
- none
- unknown

PPRC Tasks

Every PPRC action can be executed manually via the ESS Specialist Web Interface. To enable the execution of PPRC actions via command line interfaces, PPRC actions can be stored as PPRC tasks. The following PPRC task types are possible:

- Establish paths
- Remove paths
- Establish Synchronous PPRC copy pair
- Establish PPRC Extended Distance copy pair
- Suspend PPRC copy pair
- Terminate PPRC copy pair

Depending on the PPRC task type, different PPRC paths or PPRC copy options are available. Details can be found in [Cea04, p. 232].

7.5.2 Establishing PPRC Paths

The implementation focuses on fibre channel links for PPRC paths. Although ESCON links may also work, they have not been tested with this implementation.

It is recommended to use at least two different physical links for PPRC. With fibre channel, each physical link can be used in both directions. This is not possible with ESCON.

Depending on customer needs, the PPRC consistency group option can be activated during establishment of PPRC paths. This affects the behavior

in case of an outage of all PPRC paths between two Logical Sub-Systems (see section 4.6.2).

7.5.3 Necessary PPRC Tasks

Various PPRC tasks are necessary for each highly available service. In case a service uses more than one volume, the tasks must be configured for every volume and then combined into a group task. Tasks can only be created via the ESS Copy Services Web User Interface. It is not possible to automate the creation of PPRC tasks. Mandatory PPRC tasks are described in the paragraphs below:

PPRC Failover from Site A to Site B

During normal operations, volumes at the production site are in PPRC source, full-copy state. Volumes at the recovery site are in PPRC target, full-copy state. The PPRC failover task from site A to site B is necessary if the application currently uses site A as production site and the application must be switched to site B.

Nodes at site B cannot write to the PPRC target volumes. To allow write access, a PPRC failover task must be executed successfully. This brings the volumes at the recovery site (site B in this case) to PPRC source, suspended state. Volumes at the production site remain in their state. Nodes at the recovery site can mount the volumes in read/write mode. Updates to these volumes are not mirrored back to the production site. They are only marked to allow a fast resynchronization later with a PPRC failback task. More information on creating PPRC failover tasks can be found in [Cea04, p. 413 and p. 426].

PPRC Failback after PPRC Failover from Site A to Site B

No data writes are mirrored to site A after the PPRC failover to site B. If the current data at site B is valid, the mirroring to site A can be initiated with a PPRC failback task. The PPRC failback task does not allow a fail-back of the application to site A. It only ensures that volume mirroring is done back to the former production site. To do a fail-back of the application, another PPRC failover task from site B to site A is necessary after this PPRC failback task.

Before a PPRC failback task is executed it is a good idea to create point-in-time copies of the volumes at site A. If the new data at site B was erroneous, these copies can help as last known-good data sets. Without point-in-time copies, the PPRC failback overwrites these last known-good data sets. Until the PPRC failback task finishes, the volumes at site A have no data consistency at volume level. There is only one usable copy of each volume during PPRC failback.

PPRC Failover from Site B to Site A

This is the same task as “PPRC failover from site A to site B”, but with switched roles. The task is used for applications that use site B as production site and need to be switched to site A. It is also necessary for applications that use site A as production site, to allow an application fail-back after a whole application fail-over (this includes a PPRC failover and PPRC failback operation).

The execution of this task will switch volume states at site A from PPRC target, `full_copy` or `suspended` to PPRC source, `suspended`. Volumes states at site B will remain the same. Again, no data mirroring takes place after this PPRC failover task.

PPRC Failback after PPRC Failover from Site B to Site A

This task is necessary to initiate data mirroring from site A to site B after the PPRC failover from site B to site A. Again, it is wise to make a point-in-time copy at site B in this case.

Establish PPRC Connections

To initiate the PPRC mirroring between volumes, this task for establishing of PPRC connections is necessary. PPRC connections can either be configured from site A to site B or from site B to site A, depending where the PPRC source volumes should be located initially. The necessary task type is *Establish Synchronous PPRC copy pair*. The options *Copy entire volume* and *Permit read from secondary* must be set. The latter option is necessary to allow nodes at the recovery site to read partition tables at boot time.

Re-Establish PPRC Connections from Site A to Site B

After a total outage of all PPRC paths, PPRC source volumes will go to `suspended` state. When the PPRC paths are available again, PPRC connections are not resumed automatically. To resume the PPRC mirroring, PPRC connections must be re-established. Again, a task with the type *Establish Synchronous PPRC copy pair* is necessary. The options *Copy out-of-sync cylinders only* and *Permit read from secondary* are required.

This task is necessary if PPRC mirroring was done from site A to site B as the outage of the PPRC paths occurred.

Re-Establish PPRC Connections from Site B to Site A

This task has the same functionality as the previous task, with interchanged roles. It is necessary if PPRC mirroring was done from site B to site A as the outage of the PPRC paths occurred.

Terminate PPRC Connections from Site A to Site B

Sometimes it may be possible that data updates after a PPRC failover should be discarded and the old data set should be used again. This can happen if wrong data is written at the recovery site. To allow re-establishment of the PPRC connection from scratch, the old PPRC connections must be terminated. As both sites are PPRC source volumes after a PPRC failover, two tasks are necessary for this. This task terminates the PPRC connections from site A to site B. The task type is *Terminate PPRC copy pair*, with the option *Schedule task with source logical subsystem*.

Terminate PPRC Connections from Site B to Site A

This task terminates PPRC connections from site B to site A. Task type and option are the same as with the previous task.

7.6 Service-dependent PPRC Configuration Files

Every service that uses PPRC-mirrored volumes needs its own configuration file. The values assigned to the variables provide all necessary information to the `pprcvolume` script to automate the PPRC management. The configuration files must be copied to every node at both sites. An example for an NFS server is shown in figure 7.1.

siteAName, siteBName: names of the two sites. This information is used for logging purposes.

serviceName: name of the service that uses the configured volumes. The value is used for the name of the lock-file.

siteAhosts, siteBhosts: lists the names of all nodes in the cluster that may mount the configured volumes. The names must be the same as returned by the `hostname` command at the individual nodes. Host names are separated by spaces. The variables are necessary to enable the `pprcvolume` script to determine whether it is executed at site A or site B.

ipCSSA, ipCSSB: IP addresses or resolvable hostnames of Copy Services Server A and B. It is better to use IP addresses as they will also work if name resolution is impossible. If hostnames are used, it is wise to insert them with their corresponding IP addresses to `/etc/hosts` at each node.

accessESSCLI, accessCSSCLI: location of the access files for the ESSCLI (Enterprise Storage Server Command Line Interface) and the CSSCLI (Copy Services Server Command Line Interface).

```
# Configuration file for automated PPRC management
# (c) 2004 Werner Fischer, fischerw(at)at.ibm.com
# License: GPL

siteAName="Mainz"
siteBName="Frankfurt"
serviceName="nfserver"
siteAhosts="minnie mickey"
siteBhosts="moe dell"
ipCSSA="9.155.51.231"
ipCSSB="9.155.51.233"
accessESSCLI="/opt/ibm/ibm2105cli/securityFileESSCLI.cfg"
accessCSSCLI="/opt/ibm/ibm2105cli/securityFileCSSCLI.cfg"
siteAtoBfailover="EFO_GRP_B_A"
siteAtoBfailback="EFB_GRP_B_A"
siteBtoAfailover="EFO_GRP_A_B"
siteBtoAfailback="EFB_GRP_A_B"
establishPPRCConnections="ESP_GRP_A_B"
siteAtoBreEstablishPPRCConnections="RESP_GRP_A_B"
siteBtoAreEstablishPPRCConnections="RESP_GRP_A_B"
siteAterminatePPRCConnections="TSP_GRP_A_B_S"
siteBterminatePPRCConnections="TSP_GRP_B_A_S"
siteAVolumes="40028296 40128296 40228296"
siteBVolumes="40028244 40128244 40228244"
#noSyncBlockAppStart="yes"
#allowSimplexStart="yes"
```

Figure 7.1: Example configuration file `nfserver.conf`.

siteAtoBfailover: name of the task that executes a PPRC failover from site A to site B for all volumes of the configured service. If more than one volume is used, this name must refer to a group task.

siteAtoBfailback: name of the task that executes a PPRC failback after a PPRC failover from site A to site B. After this task is executed, the volumes at site B will be PPRC sources and the volumes at site A PPRC targets.

siteBtoAfailover: name of the task that executes a PPRC failover from site B to site A.

siteBtoAfailback: name of the task that executes a PPRC failback after a PPRC failover from site B to site A.

establishPPRCConnections: name of the task that establishes the initial PPRC connections. The volumes must be in PPRC simplex state to allow the first establishment of the PPRC connection.

siteAtoBreEstablishPPRCConnections: name of the task to re-establish PPRC Connections from site A to site B.

siteBtoAreEstablishPPRCConnections: name of the task to re-establish PPRC Connections from site B to site A.

siteAterminatePPRCConnections: name of the task that terminates PPRC connections with the option “schedule task with source logical subsystem” at site A.

siteBterminatePPRCConnections: name of the task that terminates PPRC connections with the option “schedule task with source logical subsystem” at site B.

siteAvolumes, siteBvolumes: volume IDs at site A and site B that will be used by the service. A volume ID contains a three digit volume number and a 5 digit ESS serial number. **siteAvolumes** and **siteBvolumes** must be configured in the same order, so that the first volume in **siteAvolumes** will be mirrored with the first volume in **siteBvolumes** and so on. Volume IDs are separated by spaces.

noSyncBlockAppStart: option to block the start of applications if both sites are not synchronous. This option is reasonable if PPRC consistency groups are used. It has no impact on PPRC failover operations. If not defined, **noSyncBlockAppStart** is set to **no**.

allowSimplexStart: allows the start of applications even if the volumes are in PPRC simplex state. This can be useful after a disaster that destroyed an ESS. In that case, the old PPRC connections should be terminated before new PPRC connections to a new ESS are established. To allow local fail-over even in those situations, **allowSimplexStart** must be configured to **yes**. This option should only be used temporarily. If not defined, it is set to **no**.

7.7 The pprcvolume Script

7.7.1 start Operation

The **start** operation is the core of the prototype. It contains the most complex functions of the prototype and ensures correct data replication states. To enable the start of applications, PPRC states must allow read/write access to the volumes. The **start** operation queries all volume states and determines if this is possible. In certain situations, it takes actions to bring

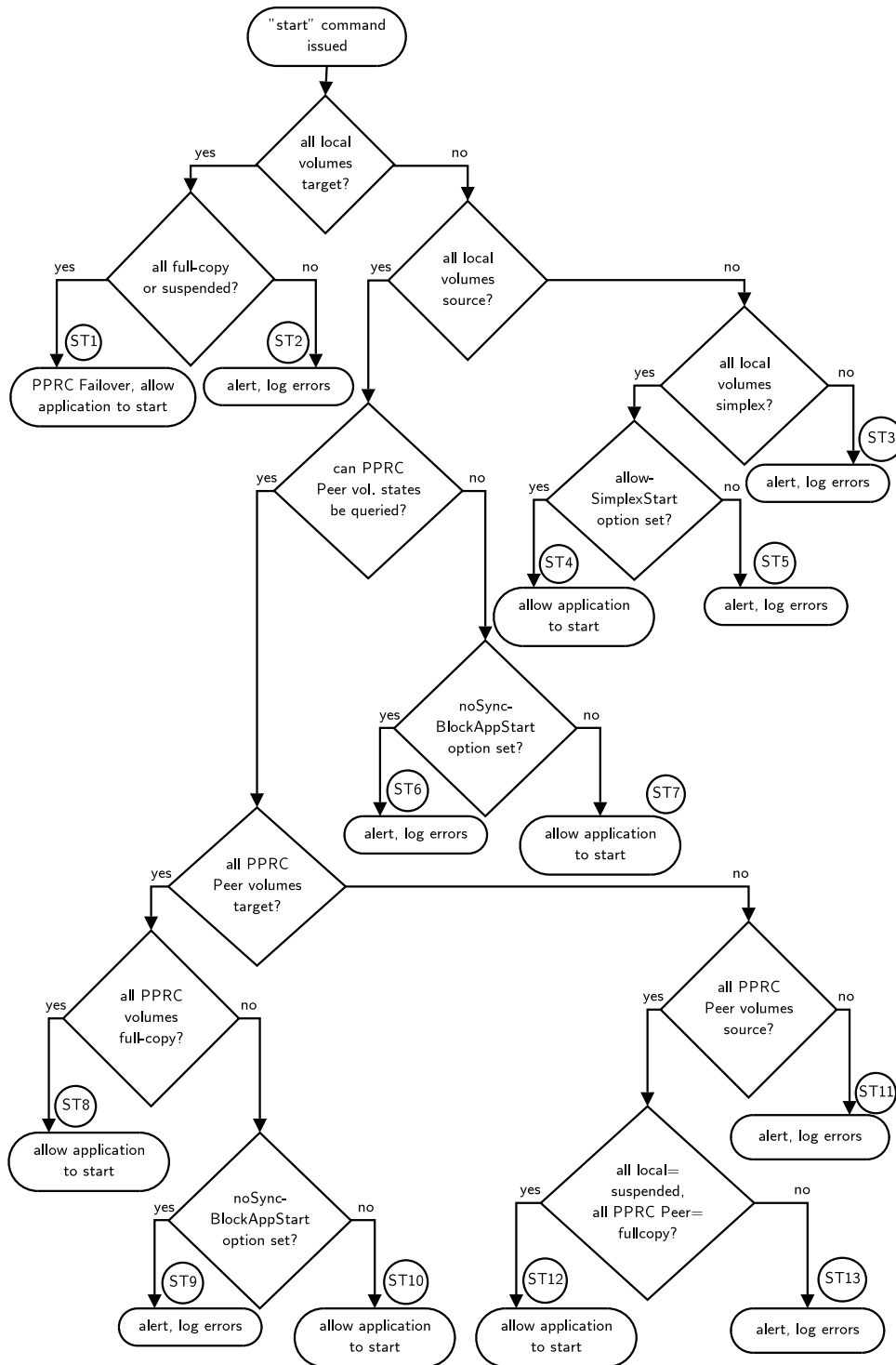


Figure 7.2: Flowchart of the start operation.

the volumes to a state that allows read/write access. If this is impossible, errors are logged and the script exits with a return code other than 0. Figure 7.2 shows the flowchart of the `start` operation. The start operation is used by the cluster manager during the startup of applications. A return code of 0 must be mandatory for the cluster manager to mount the volumes.

Are all local volumes PPRC targets?

At first, the `pprcvolume` script checks whether all local volumes are PPRC targets. PPRC target volumes can be used read-only to read the partition table during booting of a node. If the local volumes are PPRC targets in `full_copy` or `suspended` state (state ST1 in the flowchart), a PPRC failover operation must be executed. After a successful PPRC failover, former PPRC target volumes will become PPRC source volumes in `suspended` state. This allows write access to the volumes. The states of the peer volumes in the PPRC pairs remain the same.

If not all PPRC target volumes are either `full_copy` or `suspended`, no PPRC failover operation is executed and applications may not start (state ST2).

Are all local volumes PPRC sources?

If all local volumes are in PPRC source state, further investigations about the state of the PPRC peer volumes are necessary. First of all, it must be checked if the PPRC peer volume states can be queried at all. This is not possible if an ESS or its network connection fails. Depending on the `noSyncBlockAppStart` option, applications are prohibited to start (states ST6 and ST7).

Are all PPRC peer volumes targets? If the peer volumes are PPRC targets and in `full_copy` state, applications are allowed to start (state ST8). This happens for example in a local fail-over situation or at the initial start of the application.

In case that the peer volumes are PPRC targets, but not all volumes are in `full_copy` state, applications are prohibited to start depending on the `noSyncBlockAppStart` option (states ST9 and ST10). This can happen if all PPRC paths are down.

Are all PPRC peer volumes sources? If both the local and the peer volumes are PPRC sources, a PPRC failover operation may have happened before. This situation can arise when a fail-over within the recovery site is necessary after a disaster at the primary site. It is also possible that applications should be started at the production site (after the application ran at the recovery site), but no PPRC failback operation was executed before. In

the first case, the local PPRC source volumes should be in suspended state, remote PPRC source volumes in full_copy state. If this is true, applications are allowed to start (state ST12), in other cases not (ST13).

If the PPRC peer volumes have different PPRC states, applications are not allowed to start (ST11).

Are all local volumes PPRC simplex?

Depending on the `allowSimplexStart` option, applications may start (ST4) or not (ST5).

If the local volumes have different PPRC states, applications may not start (state ST3).

7.7.2 stop Operation

This operation removes the lock-file that indicates the online state of the `pprcvolume` resource. The `stop` operation is used by the cluster manager if an application is switched to another node, or the application should generally be stopped.

7.7.3 PPRCfailback Operation

A PPRC failover operation is necessary during a failover of the application to the other site. This leaves the volumes at the former production site in their current PPRC states. The volumes at the recovery site will become PPRC sources in suspended state. Depending on the reason for the site fail-over, the ESS at the production site may become available again.

The PPRC failback operation ensures a minimum re-synchronization time if new data tracks should be mirrored back to the original production site. This decision, if the new data should be mirrored back must be made by the administrator. As a consequence, this operation is for administrator's use only and must not be automatically executed by the cluster manager. Depending on the volume states at the original production site, the PPRC failback operation executes different actions.

Volumes at the Production Site are PPRC Simplex: this will happen if the ESS at the production site is replaced after a disaster.

By executing the PPRC failback, all data will be copied from the recovery to the production site.

Volumes at the Production Site are PPRC Sources in full-copy or suspended State without changed Data Tracks: this situation arises if the ESS at the production site is not destroyed.

Only out-of-sync data will be copied back to the production site. This shortens the re-synchronization time.

Volumes at the Production Site are PPRC Sources in full-copy or suspended State with changed Data Tracks: this situation should never occur, as nodes at both sites must have written to the data. This can only happen in a split brain situation, if operators grant operational quorum to both sites. This harms data consistency at application level. The data consistency at a single volume level is still preserved, But there is no chance to merge the new data of both sites.

The ESS at the recovery site (which executed the PPRC failover) will discover which data tracks have been modified at each site. All these modified data tracks will be copied back from the recovery to the production ESS. This also guarantees that changed data tracks at the former primary site will be overwritten, which is necessary to keep data consistency at volume level. All new data written to the primary site is lost.

7.7.4 status Operation

Determines the online state of the pprcvolume resource. If the lock-file exists, the resource is treated as online. Cluster managers can use the `status` operation to monitor the state of the pprcvolume resource.

7.7.5 statusReport Operation

Gives an detailed overview about the current volume states. This operation is not used by the cluster manager. It is useful for the operator to check volume states without the ESS Copy Services Web User Interface. The output of the operation is written to the log to ensure traceability. An example output is shown in figure 7.3.

7.7.6 establishPPRC Operation

Establishes the PPRC connections between the two sites. This operation can be used for the initial synchronization or after a replacement of an ESS. It may only be executed by the administrator, never by a cluster manager. It copies the entire volumes to the other site.

7.7.7 reEstablishPPRC Operation

Similar to `establishPPRC`, `reEstablishPPRC` switches to synchronous replication between the two sites. But this operation copies only out-of-sync tracks to the other site. This shortens the re-synchronization time. The operation may only be used after an outage of all PPRC paths, which causes the source volumes to go to suspended state. It may only be executed by the administrator.

```

May 4 08:07:37 pprcvolume[3031]: - performing statusReport operation -
May 4 08:07:37 pprcvolume[3032]: This host is at Mainz (site A)
May 4 08:07:39 pprcvolume[3126]: Local volume: 40028296,source,synchronous,fullcopy
May 4 08:07:41 pprcvolume[3220]: Peer volume: 40028244,target,synchronous,fullcopy
May 4 08:07:43 pprcvolume[3314]: Local volume: 40128296,source,synchronous,fullcopy
May 4 08:07:45 pprcvolume[3408]: Peer volume: 40128244,target,synchronous,fullcopy
May 4 08:07:47 pprcvolume[3502]: Local volume: 40228296,source,synchronous,fullcopy
May 4 08:07:49 pprcvolume[3596]: Peer volume: 40228244,target,synchronous,fullcopy
May 4 08:07:49 pprcvolume[3597]: volumes=3
May 4 08:07:49 pprcvolume[3598]: lSource=3
May 4 08:07:49 pprcvolume[3599]: lTarget=0
May 4 08:07:49 pprcvolume[3600]: lStatusCopy_pending=0
May 4 08:07:49 pprcvolume[3601]: lStatusSuspended=0
May 4 08:07:49 pprcvolume[3602]: lStatusFullcopy=3
May 4 08:07:49 pprcvolume[3603]: rSource=0
May 4 08:07:49 pprcvolume[3604]: rTarget=3
May 4 08:07:49 pprcvolume[3605]: rStatusCopy_pending=0
May 4 08:07:49 pprcvolume[3606]: rStatusSuspended=0
May 4 08:07:49 pprcvolume[3607]: rStatusFullcopy=3
May 4 08:07:49 pprcvolume[3608]: - statusReport operation finished -

```

Figure 7.3: Log output of statusReport operation

7.7.8 terminatePPRC Operation

In some cases it may be necessary to terminate the PPRC connections. This may be useful if an ESS must be replaced. This operation may only be used by the administrator.

7.7.9 skipPPRCfailback Operation

This operation has the same function as `terminatePPRC` (`skipPPRCfailback` is only used as a synonym). This synonym name for the same operation is chosen to highlight the second case, where this function is necessary. After a PPRC failover, changes to the data are not mirrored to the former production site. If the new data written to the recovery site is not valid, the data set of the former production site can be used again as a base to start from again (if the ESS at the former production site is not destroyed). To skip a PPRC failback, the PPRC connections must be terminated with this function. As `terminatePPRC`, `skipPPRCfailback` is only for administrator's use.

7.8 Logging Configuration

The `pprcvolume` script uses the three log security levels *alert*, *info* and *debug*. The log facility for the script is *local3*. When necessary, the log facility can be changed in the script. It is recommended to write alerts

```
local3.alert    -/var/log/pprcvolume.alert
local3.info     -/var/log/pprcvolume.info
local3.debug    -/var/log/pprcvolume.debug
```

Figure 7.4: Extract of the syslogd configuration file `/etc/syslogd.conf`.

to `/var/log/messages`. Logging information of the other two supported security levels should be written to different log files. Figure 7.4 shows an example configuration for the syslog daemon.

Chapter 8

Tests

8.1 Test Environment

The test-setup is shown in figure 8.1. The zoning configuration of the Inrange FC/9000 director simulated four SAN-switches (two SAN-switches would be located in each site).

The two sites were configured in an active/active setup in the test scenario. NFS and MySQL were used as example services. A test software

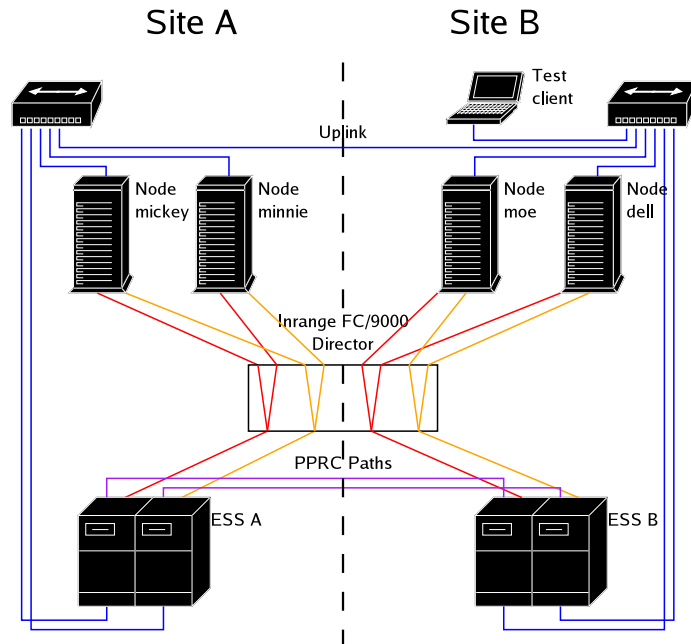


Figure 8.1: Test setup of the geographically dispersed cluster.

running on a sample client continuously wrote data to the NFS share. The sample client was connected to the LAN of site B. PPRC paths were established without consistency group option. Hence in case of an outage of all PPRC paths data writes to the PPRC source volume were allowed, although mirroring of the writes was impossible. As the test setup used a two-site configuration, manual action was sometimes necessary to execute a site fail-over.

As the tests had to be done in a isolated testing network, no connections to time servers were possible. All nodes started with synchronous time, running a local ntp daemon. Driftfiles of all nodes were calculated during one week running in a official network with access to time servers.

8.1.1 Tivoli System Automation Configuration

TSA was configured through a number of shell scripts and definition files. The configuration files for the test scenario can be found on the CD-ROM.

8.1.2 pprcvolume Configuration

The pprcvolume configuration files for the NFS and the MySQL server can be found in appendix A.

8.2 Initial Setup and Synchronization

At the beginning, all necessary PPRC tasks had to be configured and the synchronous PPRC connections manually established. This was done via the ESS Specialist Web interface.

8.3 Site A down

Expected behavior: in case of a complete outage of a site the remaining site brings up services. In a two-site configuration cluster nodes cannot distinguish between a split brain and a disaster situation. Therefore manual intervention is necessary.

Test result: Nodes at the remaining site wrote warnings to their system log files (`/var/log/messages`) that indicated the `PENDING_QUORUM` state:

```
Apr 29 15:50:21 moe ConfigRM[2546]: (Recorded using libct_ffdc.a cv2)::Error ID: ::
:Reference ID: ::Template ID: 0::Details File: ::Location: RSCT,PeerDomain.C,1.
99.5.2,12652          ::CONFIGRM_PENDINGQUORUM_ER The operational quorum state
of the active peer domain has changed to PENDING_QUORUM. This state usually indicat
es that exactly half of the nodes that are defined in the peer domain are online.
In this state cluster resources cannot be recovered although none will be stopped e
xplicitly.
Apr 29 15:50:21 moe RecoveryRM[2660]: (Recorded using libct_ffdc.a cv 2)::Error ID:
```

```

825...RUEY./wK0/7gtAo.....::Reference ID:  ::Template ID: 0::Det
ails File:  ::Location: RSCT,Protocol.C,1.13,1600          ::RECOVERYRM
_INFO_4_ST A member has left. Node number = 1
Apr 29 15:50:21 moe RecoveryRM[2660]: (Recorded using libct_ffdc.a cv 2)::Error ID:
825...RUEY./1P0/7gtAo.....::Reference ID:  ::Template ID: 0::Det
ails File:  ::Location: RSCT,Protocol.C,1.13,1600          ::RECOVERYRM
_INFO_4_ST A member has left. Node number = 2

```

Granting of operational quorum was necessary to bring up services that had been running at site A. This was done with the following command, executed at the node `moe` (the command may have been executed at any node at the remaining site B):

```
moe# runact -c IBM.PeerDomain ResolveOpQuorumTie Ownership=1
```

Nodes at site B got the quorum back:

```

Apr 29 15:55:09 moe ConfigRM[2546]: (Recorded using libct_ffdc.a cv2)::Error ID: ::
:Reference ID:  ::Template ID: 0::Details File:  ::Location: RSCT,PeerDomain.C,1.
99.5.2,12648          ::CONFIGRM_HASQUORUM_ST The operational quorum state of t
he active peer domain has changed to HAS_QUORUM. In this state, cluster resources m
ay be recovered and controlled as needed by management applications.

```

After operational quorum was granted, the NFS service was started:

```

Apr 29 15:55:11 dell pprcvolume[26041]: - performing start operation -
Apr 29 15:55:11 dell pprcvolume[26045]: This host is at Linz (site B)
...
Apr 29 15:56:44 dell /etc/saf1/nfsserver/nfsserver:[28397]: NFS server started

```

If `runact` command was executed with `Ownership=0`, operational quorum would have been denied.

8.4 ESS A down

Expected behavior: an outage of a single ESS must lead to a site fail-over. It is noticed by the cluster manager's file system monitoring script. As applications first try to fail-over to another node at the same site, the site fail-over to the other site will be executed after all local nodes have tried to bring up the application.

Test result: Nodes that were using ESS A received SCSI errors after ESS A went down:

```

Apr 29 21:35:47 minnie kernel: SCSI disk error : host 1 channel 0 id 0 lun 0 return
code = 8000002
Apr 29 21:35:47 minnie kernel: Current sd08:11: sense key Hardware Error
Apr 29 21:35:47 minnie kernel: Additional sense indicates Internal target failure
Apr 29 21:35:47 minnie kernel: I/O error: dev 08:11, sector 50512
Apr 29 21:35:47 minnie kernel: scsi(2): Waiting for LIP to complete.
Apr 29 21:35:47 minnie kernel: scsi(2): Topology - (F_Port), Host Loop address 0xfff
f
Apr 29 21:35:48 minnie kernel: qla2x00_find_all_fabric_devs GNN_FT Failed-Try issuin
g GAN
...

```

As soon as all paths of a vpath device were recognized as OFFLINE, the monitoring for this resource reported this to TSA. The NFS service was stopped. The other node at the same site (mickey) tried to bring up the nfs service twice. This failed as the ESS was not reachable. The pprcvolume script logged the reason for this.

```
Apr 29 21:36:56 mickey pprcvolume[14321]: Alert: state (ST3), apps may NOT start, exiting with RC 1
```

Finally, node dell at site B started the nfs service.

```
Apr 29 21:55:34 dell pprcvolume[21469]: - performing start operation -
Apr 29 21:55:34 dell pprcvolume[21471]: This host is at Linz (site B)
Apr 29 21:55:44 dell pprcvolume[21733]: Local volume: 40028244,target,synchronous,fullcopy
Apr 29 21:55:53 dell pprcvolume[21986]: Peer volume: 40028296,,,
...
Apr 29 21:56:30 dell pprcvolume[22995]: Alert: state (ST1), executing PPRC failover Task (EFO_GRP_B_A)
Apr 29 21:56:47 dell pprcvolume[23356]: PPRC Failover Task (EFO_GRP_B_A) exited with signal 0 and returned: rsExecuteTask: Command successful
Apr 29 21:56:47 dell pprcvolume[23357]: Info: Apps may start, exiting with RC 0
Apr 29 21:56:47 dell pprcvolume[23359]: - start operation finished -
```

8.5 One Node at Site A down

Expected behavior: another node at the same site automatically starts the applications that have been running on the failed node.

Test result: The node minnie (running the nfs server) failed (was powered off). The other node at site A (mickey) started the nfs server.

```
Apr 29 16:34:46 mickey RecoveryRM[12836]: (Recorded using libct_ffdc.a cv 2)::Error ID: 825...48FY./sRDO7gtAu.....::Reference ID: ::Template ID: 0:::Details File: ::Location: RSCT,Protocol.C,1.13,1600 ::RECOVERM_INFO_4_ST A member has left. Node number = 1
Apr 29 16:34:53 mickey hatsd[12558]: hadms: Loading watchdog using command: insmod softdog soft_margin=8 nowayout=0.
Apr 29 16:34:53 mickey kernel: Software Watchdog Timer: 0.05, timer margin: 8 sec
Apr 29 16:34:53 mickey pprcvolume[26649]: - performing start operation -
...
Apr 29 16:35:07 mickey pprcvolume[27420]: Info: state (ST8), all volumes full_copy, apps may start, exiting with RC 0
Apr 29 16:35:07 mickey pprcvolume[27421]: - start operation finished -
...
Apr 29 16:35:25 mickey /etc/safl/nfsserver/nfsserver:[27846]: NFS server started
```

After the defective node was powered on again, the nfs service continued to run on mickey.

8.6 Network Connection between Site A and Site B down

Expected behavior: all running applications continue to run on their current nodes. The outage of the network connection between the two sites results in two sub-clusters. Each of them will go to `PENDING_QUORUM` state. If a site fail-over is necessary (e.g. from site A to site B), quorum must be denied for site A before the site B gets operational quorum. Nodes at site A that are running services will be forced to reboot immediately after quorum is denied through the `runact` command. After granting operational quorum at site B, nodes at site B will bring up services that have been running on site A. When the network connection is up again, a PPRC fail-back can be executed to mirror the new data back to site A. During this re-synchronization, the data at site A is not consistent.

Test result: All nodes were still up and running. The cluster split up into two sub-clusters. The output of the `lsrpnode` command showed the operational state of the nodes in the sub-clusters (the output depended on the node where the command was executed):

```
mickey:~ # lsrpnode
Name OpState RSCTVersion
mickey Online 2.3.2.2
dell Offline 2.3.2.2
minnie Online 2.3.2.2
moe Offline 2.3.2.2
mickey:~ #
```

```
moe:~ # lsrpnode
Name OpState RSCTVersion
mickey Offline 2.3.2.2
dell Online 2.3.2.2
minnie Offline 2.3.2.2
moe Online 2.3.2.2
moe:~ #
```

As the test client could not reach nodes at site A after the network outage, a site fail-over from site A to site B was necessary. This was done by denying operational quorum at site A and granting operational quorum at site B.

```
minnie# runact -c IBM.PeerDomain ResolveOpQuorumTie Ownership=0
moe# runact -c IBM.PeerDomain ResolveOpQuorumTie Ownership=1
```

After the `runact` command was executed on `minnie`, this node rebooted immediately. The other node at site A (`mickey`) logged information about the loss of quorum:

```
Apr 29 13:01:23 mickey ConfigRM[20140]: (Recorded using libct_ffdc.a cv 2):::Error I
D: :::Reference ID: :::Template ID: 0:::Details File: :::Location: RSCT,PeerDomain
```

```
.C,1.99.5.2,12656          :::CONFIGRM_NOQUORUM_ER The operational quorum state
of the active peer domain has changed to NO_QUORUM. This indicates that recovery of
cluster resources can no longer occur and that the node may be rebooted or halted
in order to ensure that critical resources are released so that they can be recover
ed by another sub-domain that may have operational quorum.
```

After the sub-cluster at site B got operational quorum, a node at site B started the NFS server.

```
Apr 29 13:03:40 dell ConfigRM[20097]: (Recorded using libct_ffdc.a cv 2)::Error ID:
:::Reference ID: :::Template ID: 0:::Details File: :::Location: RSCT,PeerDomain.C
,1.99.5.2,12648          :::CONFIGRM_HASQUORUM_ST The operational quorum state o
f the active peer domain has changed to HAS_QUORUM. In this state, cluster resource
s may be recovered and controlled as needed by management applications.
Apr 29 13:03:41 dell hatsd[20024]: hadms: Loading watchdog using command: insmod sof
tdog soft_margin=8 nowayout=0.
Apr 29 13:03:41 dell kernel: Software Watchdog Timer: 0.05, timer margin: 8 sec
Apr 29 13:03:41 dell pprcvolume[21742]: - performing start operation -
Apr 29 13:03:41 dell pprcvolume[21743]: This host is at Linz (site B)
...
Apr 29 13:04:37 dell pprcvolume[23256]: Alert: state (ST1), executing PPRC failover
Task (EFO_GRP_B_A)
Apr 29 13:04:55 dell pprcvolume[23610]: PPRC Failover Task (EFO_GRP_B_A) exited with
signal 0 and returned: rsExecuteTask: Command successful
Apr 29 13:04:55 dell pprcvolume[23611]: Info: Apps may start, exiting with RC 0
Apr 29 13:04:55 dell pprcvolume[23613]: - start operation finished -
...
Apr 29 13:05:15 dell /etc/saf1/nfssserver/nfssserver:[24101]: NFS server started
```

As the new data at site B was valid, a PPRC failback was executed.

```
Apr 29 13:19:32 moe pprcvolume[8692]: - performing failback operation -
Apr 29 13:19:32 moe pprcvolume[8694]: This host is at Linz (site B)
...
Apr 29 13:20:40 moe pprcvolume[10668]: PPRC Failback Task (EFB_GRP_B_A) exited with
signal 0 and returned: rsExecuteTask: Command successful
```

8.7 PPRC between ESS A and ESS B down

Expected behavior: as consistency groups are not activated, services continue to run on their current nodes. If data is written to the PPRC source volumes, the state of this PPRC sources changes from full-copy to suspended. PPRC target volumes will remain in full-copy state, even if new data is written to the associated PPRC source volumes.

Test result: After all PPRC paths failed, nodes that wrote to volumes got SCSI errors:

```
Apr 29 14:58:27 minnie kernel: SCSI disk error : host 2 channel 0 id 0 lun 2 return
code = 8000002
Apr 29 14:58:27 minnie kernel: Current sd08:81: sense key Hardware Error
Apr 29 14:58:27 minnie kernel: Additional sense indicates Internal target failure
Apr 29 14:58:27 minnie kernel: I/O error: dev 08:81, sector 1649600
...
```

The PPRC state of the PPRC source volumes changed after they received write commands. To document this, a `pprcvolume statusReport` operation was executed manually:

```
Apr 29 15:26:18 minnie pprcvolume[29578]: - performing statusReport operation -
Apr 29 15:26:18 minnie pprcvolume[29584]: This host is at Mainz (site A)
Apr 29 15:26:20 minnie pprcvolume[29690]: Local volume: 40028296,source,synchronous,
suspended
Apr 29 15:26:22 minnie pprcvolume[29802]: Peer volume: 40028244,target,synchronous,
fullcopy
Apr 29 15:26:24 minnie pprcvolume[29952]: Local volume: 40128296,source,synchronous,
suspended
Apr 29 15:26:26 minnie pprcvolume[30056]: Peer volume: 40128244,target,synchronous,
fullcopy
Apr 29 15:26:28 minnie pprcvolume[30206]: Local volume: 40228296,source,synchronous,
suspended
Apr 29 15:26:30 minnie pprcvolume[30300]: Peer volume: 40228244,target,synchronous,
fullcopy
...
Apr 29 15:26:30 minnie pprcvolume[30312]: - statusReport operation finished -
```

8.8 Split Brain between Site A and Site B

Expected behavior: Services continue to run at both sites. If additional cluster resources fail, they will not be recovered as no site has quorum.

Test result: All nodes wrote warnings to `/var/log/messages` that the operational quorum state changes from `HAS_QUORUM` to `PENDING_QUORUM` (compare section 8.3).

8.9 Network at Site A down

Expected behavior: Active nodes at site A that are running services reboot immediately. Services at site A will be down. Manual action is necessary to bring up these services at site B.

Test result: The node `minnie` rebooted immediately as it had been running the NFS service. The other node at site A, `mickey`, continued to run but logged information about the loss of quorum to `/var/log/messages`. After executing of the `runact` command with the parameter `Ownership=1` at `moe` (any node at site B would have been possible), the node `dell` brought up the NFS service.

```
Apr 29 17:00:34 dell ConfigRM[1984]: (Recorded using libct_ffdc.a cv 2):::Error ID:
:::Reference ID: :::Template ID: 0:::Details File: :::Location: RSCT,PeerDomain.C,
1.99.5.2,12648      :::CONFIGRM_HASQUORUM_ST The operational quorum state of
the active peer domain has changed to HAS_QUORUM. In this state, cluster resources
may be recovered and controlled as needed by management applications.
Apr 29 17:00:34 dell pprcvolume[19094]: - performing start operation -
Apr 29 17:00:34 dell pprcvolume[19095]: This host is at Linz (site B)
```

```
...
Apr 29 17:02:08 dell /etc/saf1/nfssserver/nfssserver:[21430]: NFS server started
```

8.10 Network completely down

Expected behavior: Nobody can use any services as the network is completely down. The cluster will split up into four sub-clusters (each node will build a sub-cluster). Nodes that have been running services reboot immediately. Other nodes continue to run and indicate the loss of quorum. All services will be stopped. Data consistency is still preserved.

Test result: The nodes `mickey` and `moe` rebooted, as they had been running services. The two other nodes remained up and indicated the loss of quorum. All services went down.

8.11 Rolling Disaster Situation 1

In the simulated example, the rolling disaster leads to outages in the following order:

1. LAN connection of the ESS at site A fails.
2. LAN connection of node `minnie` fails two minutes later.
3. All other components at site A fail 10 minutes later.

Expected behavior: The outage of the LAN connection of ESS A causes no problems as long as no local fail-over is necessary. Immediately after the LAN connection of `minnie` fails, it reboots itself to protect shared resources. The other node at site A, `mickey`, tries to bring up the NFS service. As the PPRC state of the volumes of ESS A cannot be queried, the startup of the NFS service at `mickey` fails. After that, another node at site B brings up the NFS service. When `mickey` also fails as all remaining components at site A fail, the quorum state changes to `PENDING_QUORUM`.

Test result: `minnie` rebooted after its LAN connection failed. The other node at site A (`mickey`) tried to bring up the NFS service:

```
Apr 30 07:00:02 mickey RecoveryRM[22956]: (Recorded using libct_ffdc.a cv 2)::Error
  ID: 825....GpRY./otC.7gtAu.....:Reference ID: :::Template ID: 0::
:Details File: :::Location: RSCT,Protocol.C,1.13,1600          :::RECOVE
RYRM_INFO_4_ST A member has left. Node number = 1
Apr 30 07:00:02 mickey RecoveryRM[22956]: (Recorded using libct_ffdc.a cv 2)::Error
  ID: 825....GpRY./cxC.7gtAu.....:Reference ID: :::Template ID: 0::
:Details File: :::Location: RSCT,Protocol.C,1.13,1635          :::RECOVE
RYRM_INFO_6_ST Master has left, a new master has taken over. Node number of the new
master = 4
Apr 30 07:00:04 mickey hatsd[19047]: hadms: Loading watchdog using command: insmod s
oftdog soft_margin=8 nowayout=0.
```

```

Apr 30 07:00:04 mickey kernel: Software Watchdog Timer: 0.05, timer margin: 8 sec
Apr 30 07:00:04 mickey pprcvolume[31441]: - performing start operation -
Apr 30 07:00:04 mickey pprcvolume[31442]: This host is at Mainz (site A)
Apr 30 07:00:12 mickey pprcvolume[31682]: Local volume: 40028296,,,
Apr 30 07:00:12 mickey pprcvolume[31705]: Peer volume: ,,,
Apr 30 07:00:20 mickey pprcvolume[31921]: Local volume: 40128296,,,
Apr 30 07:00:20 mickey pprcvolume[31944]: Peer volume: ,,,
Apr 30 07:00:28 mickey pprcvolume[32192]: Local volume: 40228296,,,
Apr 30 07:00:28 mickey pprcvolume[32215]: Peer volume: ,,,
...
Apr 30 07:00:28 mickey pprcvolume[32227]: Alert: state (ST3), apps may NOT start, exiting with RC 1
Apr 30 07:00:28 mickey pprcvolume[32228]: - start operation finished -

```

As *mickey* could not start the NFS service, but was still part of the cluster, another node at site B brought up the NFS service without manual intervention.

```

Apr 30 07:10:48 dell pprcvolume[6752]: - performing start operation -
Apr 30 07:10:48 dell pprcvolume[6757]: This host is at Linz (site B)
...
Apr 30 07:11:43 dell pprcvolume[8255]: Alert: state (ST1), executing PPRC failover Task (EFO_GRP_B_A)
Apr 30 07:12:01 dell pprcvolume[8646]: PPRC Failover Task (EFO_GRP_B_A) exited with signal 0 and returned: rsExecuteTask: Command successful
Apr 30 07:12:01 dell pprcvolume[8647]: Info: Apps may start, exiting with RC 0
Apr 30 07:12:01 dell pprcvolume[8649]: - start operation finished -
...
Apr 30 07:12:22 dell /etc/saf1/nfsserver/nfsserver:[9132]: NFS server started

```

8.12 Rolling Disaster Situation 2

In this simulated example, the rolling disaster leads to outages in the following order:

1. LAN connection of the ESS at site A fails.
2. All PPRC links fail one minute later.
3. Node *minnie* fails two minutes later (loses all its power links).

Expected behavior: The outage of the LAN connection of ESS A causes no problems as long as no local fail-over is necessary. The outage of the PPRC links also does not disrupt the running services. After *minnie* fails, the other node at site A (*mickey*) tries to bring up the NFS service. This will fail, as the state of ESS A volumes cannot be queried. Finally, the NFS service will be started at *dell*, causing a PPRC Failover from site A to site B.

Test result: After *minnie* failed, *mickey* was able to bring up the NFS service. This was not the expected behavior. The *pprcvolume* script got wrong state information about volume states of ESS A:

```

Apr 30 08:03:13 mickey pprcvolume[17081]: - performing start operation -
Apr 30 08:03:13 mickey pprcvolume[17082]: This host is at Mainz (site A)
Apr 30 08:03:22 mickey pprcvolume[17284]: Local volume: 40028296,source,synchronous,
fullcopy
Apr 30 08:03:30 mickey pprcvolume[17538]: Peer volume: 40028244,target,synchronous,
fullcopy
Apr 30 08:03:38 mickey pprcvolume[17794]: Local volume: 40128296,source,synchronous,
fullcopy
Apr 30 08:03:47 mickey pprcvolume[18009]: Peer volume: 40128244,target,synchronous,
fullcopy
Apr 30 08:03:55 mickey pprcvolume[18269]: Local volume: 40228296,source,synchronous,
fullcopy
Apr 30 08:04:03 mickey pprcvolume[18529]: Peer volume: 40228244,target,synchronous,
fullcopy
...
Apr 30 08:04:03 mickey pprcvolume[18546]: Info: state (ST8), all volumes full_copy,
apps may start, exiting with RC 0
Apr 30 08:04:03 mickey pprcvolume[18547]: - start operation finished -
...
Apr 30 08:04:20 mickey /etc/safl/nfssserver/nfssserver:[18967]: NFS server started

```

The script got information about all volume states from ESS B. Unfortunately, ESS B cached the volume states from ESS A. As the LAN connection of ESS A broke first, followed by an outage of the PPRC links, ESS A could not send updated volume states to ESS B. A manually initiated `pprcvolume statusReport` operation showed the correct state information two minutes later:

```

Apr 30 08:05:02 mickey pprcvolume[19505]: - performing statusReport operation -
Apr 30 08:05:02 mickey pprcvolume[19506]: This host is at Mainz (site A)
Apr 30 08:05:10 mickey pprcvolume[19763]: Local volume: 40028296,,,
Apr 30 08:05:10 mickey pprcvolume[19786]: Peer volume: ,,,
Apr 30 08:05:18 mickey pprcvolume[20004]: Local volume: 40128296,,,
Apr 30 08:05:18 mickey pprcvolume[20027]: Peer volume: ,,,
Apr 30 08:05:26 mickey pprcvolume[20269]: Local volume: 40228296,,,
Apr 30 08:05:26 mickey pprcvolume[20292]: Peer volume: ,,,
...
Apr 30 08:05:26 mickey pprcvolume[20304]: - statusReport operation finished -

```

There was no information about PPRC peer volumes, because the local PPRC states could not be queried (and the information about the corresponding PPRC peer volumes comes from this query).

To prevent such situations, it is necessary to set a timeout value before volume states are queried during a `pprcvolume start` operation. The timeout value depends on the maximum possible time an ESS caches volume state information.

Also without a timeout value, data consistency will never be violated. In the worst case, services will be started on another node.

Chapter 9

Conclusions

9.1 Conclusions

1. A prototype for the automation of storage subsystem functions for data replication has been developed. The implementation is shown with Tivoli System Automation for Linux as an example of a cluster manager. The prototype itself works with any Linux cluster manager that fulfills the requirements mentioned in section 7.2.
2. The prototype supports active-active configurations, two- and three-site setups (third site only for quorum) and allows each service to be individually configured.
3. The implemented prototype meets all test conditions and handles disaster situations correctly.
4. If data writes are only allowed if the mirroring of the data works correctly, automation of the replication function for cluster manager integration is very complicated (as shown in section 4.6).

9.2 Possibilities of Future Enhancements

As mentioned in [KP03, p. 7], some companies are giving serious consideration to a three-site strategy, each capable of running applications. This kind of design could allow automatic continuation of services after a disaster, even if data must be replicated to a second site to allow applications to run. Different applications may run at any of the three sites. During normal operations, data is mirrored between all sites. In case of a disaster at one site, the remaining two sites start the application that ran at the destroyed site. Data mirroring continues between the two remaining sites. The automation of data replication in such a configuration would be rather complex and will need future research.

Another nice feature would be a support for multiple ESSes per site. The current prototype could support multiple ESSes, but for a single service the maximum is one ESS per site.

The implemented prototype supports synchronous replication only. Additional support for asynchronous replication would allow the use of the implementation also for longer distances.

Appendix A

Configuration Files

A.1 pprcvolume Configuration for nfsserver

```
# Configuration file for automated PPRC management
# (c) 2004 Werner Fischer, fischerw(at)at.ibm.com
# License: GPL

# Site names of site A and site B
siteAName="Mainz"
siteBName="Linz"

# Name of the service that uses the volumes configured by this file.
serviceName="nfsserver"

# Hostnames of hosts on site A and on site B. These names must be the same names
# as returned by the hostname command on the individual hosts. Hostnames are
# separated by spaces.
# Attention: these parameters are essential as they are used to determine
#           whether the current host belongs to site A or B. If, for example,
#           the current host belongs to site A, the volumes on site A must be
#           configured as PPRC source if the host wants to access the volumes.
siteAhosts="minnie mickey"
siteBhosts="moe dell"

# IP addresses or resolveable hostnames of copy services server A and B.
ipCSSA="9.155.51.231"
ipCSSB="9.155.51.233"

# Location of security files for the ESS Storage Management CLI
# (accessESSCLI) and the ESS Copy Services CLI (accessCSSCLI).
# Attention: give full path names.
accessESSCLI="/opt/ibm/ibm2105cli/securityFileESSCLI.cfg"
accessCSSCLI="/opt/ibm/ibm2105cli/securityFileCSSCLI.cfg"

# Task names for failover and failback tasks for both directions (from site A
# to site B and from site B to site A).
siteAtoBfailover="EFO_GRP_B_A"
siteAtoBfailback="EFB_GRP_B_A"
siteBtoAfailover="EFO_GRP_A_B"
siteBtoAfailback="EFB_GRP_A_B"

# Task names for terminating PPRC connections
siteAterminatePPRC="TSP_GRP_A_B_S"
siteBterminatePPRC="TSP_GRP_B_A_S"
```

```

# Volume IDs on site A and site B that the application will use.
# Attention: all these volumes must be included in the failover and failback
#           tasks above - otherwise the volumes won't be mirrored correctly
#           by PPRC.
# The volumes must be configured in the same order, so that the first volume
# in siteAvolumes will be mirrored with the first volume in siteBvolumes, the
# second with the second and so on.
siteAVolumes="40028296 40128296 40228296"
siteBVolumes="40028244 40128244 40228244"

# This option can be useful when using consistency groups in PPRC paths. In case
# the remote ESS (PPRC Peer) is not reachable, and all local volumes are PPRC
# sources, this option prevents applications to start. This reaction can be
# desirable if you only want to start applications if data can be mirrored to
# the other site except for disaster failover situations.
# Note: this option does not influence the behavior in case of a failover
#       operation. If the local volumes are PPRC target (which is the case at
#       secondary site in case of a disaster at the primary site) applications
#       are of course allowed to start after a successful PPRC failover
#       operation.
#noSyncBlockAppStart="yes"

# This options allows the start of services even if no PPRC is set up and the
# local volumes are in PPRC simplex mode. The default behavior is to refuse the
# start of services if PPRC is not set up.
#allowSimplexStart="yes"

```

A.2 pprcvolume Configuration for mysql

```

# Configuration file for automated PPRC management
# (c) 2004 Werner Fischer, fischerw(at)at.ibm.com
# Licence: GPL

# Sitenames of site A and site B
siteAName="Mainz"
siteBName="Linz"

# Name of the service that uses the volumes configured by this file.
serviceName="mysql"

# Hostnames of hosts on site A and on site B. These names must be the same names
# as returned by the hostname command on the individual hosts. Hostnames are
# seperated by spaces.
# Attention: these parameters are essential as they are used to determine
#           whether the current host belongs to site A or B. If, for example,
#           the current host belongs to site A, the volumes on site A must be
#           configured as PPRC source if the host wants to access the volumes.
siteAhosts="minnie mickey"
siteBhosts="moe dell"

# IP addresses or resolveable hostnames of copy services server A and B.
ipCSSA="9.155.51.231"
ipCSSB="9.155.51.233"

# Location of security files for the ESS Storage Management CLI
# (accessESSCLI) and the ESS Copy Services CLI (accessCSSCLI).
# Attention: give full path names.
accessESSCLI="/opt/ibm/ibm2105cli/securityFileESSCLI.cfg"
accessCSSCLI="/opt/ibm/ibm2105cli/securityFileCSSCLI.cfg"

```

```
# Task names for failover and failback tasks for both directions (from site A
# to site B and from site B to site A).
siteAtoBfailover="EFO_GRP2_B_A"
siteAtoBfailback="EFB_GRP2_B_A"
siteBtoAfailover="EFO_GRP2_A_B"
siteBtoAfailback="EFB_GRP2_A_B"

# Task names for terminating PPRC connections
siteAterminatePPRC="TSP_GRP2_A_B_S"
siteBterminatePPRC="TSP_GRP2_B_A_S"

# Volume IDs on site A and site B that the application will use.
# Attention: all these volumes must be included in the failover and failback
# tasks above - otherwise the volumes won't be mirrored correctly
# by PPRC.
# The volumes must be configured in the same order, so that the first volume
# in siteAvolumes will be mirrored with the first volume in siteBvolumes, the
# second with the second and so on.
siteAVolumes="40328296 40428296"
siteBVolumes="40328244 40428244"

# This option can be useful when using consistency groups in PPRC paths. In case
# the remote ESS (PPRC Peer) is not reachable, and all local volumes are PPRC
# sources, this option prevents applications to start. This reaction can be
# desirable if you only want to start applications if data can be mirrored to
# the other site except for disaster failover situations.
# Note: this option does not influence the behaviour in case of a failover
# operation. If the local volumes are PPRC target (which is the case at
# secondary site in case of a disaster at the primary site) applications
# are of course allowed to start after a successful PPRC failover
# operation.
#noSyncBlockAppStart="yes"

# This options allows the start of services even if no PPRC is set up and the
# local volumes are in PPRC simplex mode. The default behavior is to refuse the
# start of services if PPRC is not set up.
#allowSimplexStart="yes"
```

Appendix B

Contents of CD-ROM

File System: Joliet

Mode: Single-Session

B.1 Diploma Thesis

Path: /

da.dvi	diploma thesis (DVI-File)
da.pdf	diploma thesis (PDF-File)
da.ps	diploma thesis (PostScript-File)

B.2 *LaTeX*-Files

Path: /latex/

da.tex	main document
0_preface.tex	Preface
0_kurzfassung.tex	Kurzfassung
0_abstract.tex	Abstract
1_introduction.tex	chapter 1
2_background.tex	chapter 2
3_disasterRecovery.tex	chapter 3
4_dataReplication.tex	chapter 4
5_examples.tex	chapter 5
6_problemStatement.tex	chapter 6
7_implementation.tex	chapter 7
8_tests.tex	chapter 8
9_conclusions.tex	chapter 9

appendix_a.tex appendix A (Configuration Files)
appendix_b.tex appendix B (Contents of CD-ROM)

B.3 Implementation

Path: /implementation/

pprcvolume pprcvolume-script
example.conf example configuration

B.4 Test configuration

Path: /testconfiguration/

mysql/ directory containing mysql configuration
nfserver/ directory containing nfserver configuration
nfserver_mysql.scenario scenario file for TSA

B.5 Bibliography

Path: /bibliography/

books/ directory containing books available in pdf
format
linux-2.4.26/ directory containing Linux 2.4.26 kernel
source
manuals/ directory containing manuals available in pdf
format
redbooks/ directory containing IBM redbooks
whitepapers/ directory containing whitepapers available in
pdf format

Bibliography

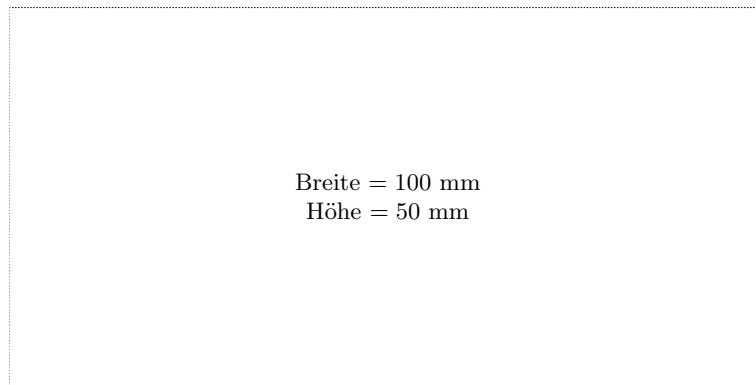
- [BK03] Jean S. Bozman and Dan Kusnetzky. Planning for Business Continuity: HP-UX and Geographically Dispersed Clusters. Technical report, International Data Corporation, Framingham, MA, USA, April 2003. IDC document 03C3670, online available at ftp://ftp.hp.com/pub/enterprise/bus_con_IDC_whitepaper.pdf.
- [Cah03] Ben Cahill. Whatis OpenDLM. <http://opendlm.sourceforge.net/docs.php>, 2003. copy on CD-ROM.
- [Cea02] Guesavo Castets et. al. *IBM TotalStorage Enterprise Storage Server Model 800*. International Business Machines Corporation, International Technical Support Organization, San Jose, CA, USA, second edition, October 2002. Redbook SG24-6424-01, copy on CD-ROM.
- [Cea04] Guesavo Castets et. al. *Implementing ESS Copy Services in Open Environments*. International Business Machines Corporation, International Technical Support Organization, San Jose, CA, USA, fourth edition, February 2004. Redbook SG24-5757-03, copy on CD-ROM.
- [Dav00] Thomas Davis. Linux 2.4.26 kernel source, /documentation/networking/bonding.txt. <http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.26.tar.bz2>, 2000. copy on CD-ROM.
- [Fre02] David Freund. Disaster Tolerant Unix: Removing the Last Single Point of Failure. Technical report, Illuminata, Inc., Nashua, NH, USA, August 2002. Online available at <http://h71000.www7.hp.com/openvms/whitepapers/Illuminata.pdf>.
- [Hew03] Hewlett-Packard Co., Palo Alto, CA, USA. *Arbitration For Data Integrity in ServiceGuard Clusters*, November 2003. Manufacturing Part Number: B7660-90078.
- [Hew04] Hewlett-Packard Co., Palo Alto, CA, USA. *Designing Disaster Tolerant High Availability Clusters*, March 2004. Manufacturing Part Number: B7660-90014.

- [Ins04] Institute of Electrical and Electronics Engineers, Inc., Piscataway, NJ, USA. *IEEE Standard 802.1w-2001 for Local and metropolitan area networks. Common specifications. Part 3: Media Access Control (MAC) Bridges, Amendment 2: Rapid Reconfiguration*, 2004. online available at <http://standards.ieee.org/getieee802/download/802.1w-2001.pdf>.
- [Int03a] International Business Machines Corporation, Armonk, New York, USA. *HACMP Remote Copy: ESS PPRC Guide*, July 2003. Publication Number SC23-4863-00, online available at <http://publibfp.boulder.ibm.com/epubs/pdf/c2348630.pdf>.
- [Int03b] International Business Machines Corporation, Armonk, New York, USA. *IBM TotalStorage Enterprise Storage Server Command-Line Interfaces User's Guide*, November 2003. Publication Number SC26-7494-03, online available at <http://publibfp.boulder.ibm.com/epubs/pdf/f2bc1i03.pdf>, copy on CD-ROM.
- [Kea04] George Kozakos et. al. *Implementing ESS Copy Services with IBM zSeries*. International Business Machines Corporation, International Technical Support Organization, San Jose, CA, USA, February 2004. Redbook SG24-5680-03, copy on CD-ROM.
- [KP03] R. F. Kern and V. T. Peltz. IBM Storage Infrastructure for Business Continuity. International Business Machines Corporation, Armonk, Now York, USA, Whitepaper, December 2003.
- [Mit04] Christoph Mitasch. Server-Based Wide Area Data Replication for Disaster Recovery. Master's thesis, Fachhochschule Hagenberg, Studiengang Computer- und Mediensicherheit, Hagenberg, Austria, 2004.
- [MS03] Evan Marcus and Hal Stern. *Blueprints for High Availability*. Wiley Publishing, Indianapolis, IN, USA, second edition, 2003.
- [Rei00] Philipp Reisner. Festplattenspiegelung übers Netzwerk für die Realisierung hochverfügbarer Server unter Linux. Master's thesis, Technische Universität Wien, Institut für Computersprachen, Vienna, Austria, May 2000.
- [Rob01] Alan Robertson. Resource fencing using STONITH. http://linux-ha.org/heartbeat/talks/ResourceFencing_Stonith.pdf, 2001.
- [Str02] Hartmut Streppel. Capmus Clusters Based on Sun Cluster 3.0 Software. Technical report, Sun Microsystems, Inc., Santa Clara, CA, USA, November 2002. Part Number 817-0369-10, online available at <http://www.sun.com/solutions/blueprints/1102/817-0369.pdf>.

- [Sun04] Sun Microsystems, Inc., Santa Clara, CA, USA. *Sun Cluster 3.x Hardware Administration Manual for Solaris OS*, January 2004. Part Number 817-0168-10, online available at <http://docs-pdf.sun.com/817-0168/817-0168.pdf>.
- [TE03] Ulf Troppens and Rainer Erkens. *Speichernetze - Grundlagen und Einsatz von Fibre Channel SAN, NAS, iSCSI und InfiniBand*. dpunkt.verlag, Heidelberg, Germany, 2003.
- [Wea02] Richard Wilkins et al. Disaster tolerant wolfpack geo-clusters. In *Proceedings of the IEEE Internal Conference on Cluster Computing (CLUSTER'02)*, Piscataway, NJ, USA, 2002. Institute of Electrical and Electronics Engineers, Inc.
- [Wea04] Cathy Warrick et. al. *IBM TotalStorage Solutions for Disaster Recovery*. International Business Machines Corporation, International Technical Support Organization, San Jose, CA, USA, January 2004. Redbook SG24-6547-01, copy on CD-ROM.
- [You97] Eric Youngdale. Linux 2.4.26 kernel source, /drivers/scsi/scsi_queue.c. <http://www.kernel.org/pub/linux/kernel/v2.4/linux-2.4.26.tar.bz2>, 1997. copy on CD-ROM.

Messbox zur Druckkontrolle

— Druckgröße kontrollieren! —



— Diese Seite nach dem Druck entfernen! —