Troubleshooting Linux Storage

A systems administrator guide

Erwin van Londen

Table of Content

List of Figures

List of Tables

Introduction

The can of worms, pandora's box, hornets nest, snake in the grass, boil the ocean. Now that we have the idioms out of the way here the first statement: *"There are hundreds of way to troubleshoot storage related issues and this book is not a silver bullet to solve all your problems!"*. This book is also not a deep-dive into the architecture or code of every kernel piece you can find in relation to storage. That would really be like boiling the ocean.
Neither is this book a guide into designing end-to-end storage infrastructures. There is a great deal of knowledge and experience required in order to get application IO behaviour in line with storage capabilities. Every piece of hardware and software from application to spindle is an important piece of the puzzle and having even one of them wrong may have a very severe impact not only on the data-path itself but also on other shared parts of that data-path.

What this book will do is provide you a methodology into troubleshooting scenario's that as a result of storage relate infrastructure problem either connected to local or SAN attached storage. I will not touch on a few things like for example FCoE from a technology side, as these may divert into a rabbit hole. I may give these technologies more attention in later articles or updated versions of the book. I'm not saying these are bad technologies but I have to make a start somewhere. Another topic that is not covered in-depth is performance. There are tips and tricks throughout the book that will or may improve your overall storage infrastructure and therefore also performance. Although I do understand that performance is always a hot topic it would go too far to incorporate it in this book at this stage.

How the book is written

I tried to write the entire book in ascii text. Mainly because I don't like word processors and secondly I wanted to be able to use a version control system so corrections and adjustments can be easily made without any hassle.

The below tools and methodologies were used:

Table 1: Tools for writing the book.

Tool	Method
Main writing tool	MS Visual Studio Code on Fedora
Format	Pandoc MarkDown
Images	PlantUML (as much as possible)
Build tool	Pandoc
Text Compiler	Latex
Revision Control	GIT
References	Zotero with BetterBibTex extension
Citations	IEEE with URL

In certain system output sections you will see that a line sometimes ends with a "\". This means that the output on the following line is actually part of this one but due to exceeding the page margins these are wrapped.

References are either internal or external to the book. Internal references are all numbered and linked in the digital edition. External references are shown in the "References" chapter at the end of the book. In the digital edition these are also hyperlinked. Short explanation references are shown as a footnote.

Thank you

I would like to thank all the people who have helped me during my career of ~25 years in the storage business. The list of names is too long to mention here but I think you know who you are when you read this. There are a few I would like to mention though.

- Horst Truestedt - The godfather of Fibre Channel and my instructor and mentor for many years.
- Declan O'Mahony and Bernd Falkenstein and all the people from Onsite Computing, which unfortunately no longer exists, for providing excellent storage training courses when I started back in the late 90-ties.
- Adrian DeLuca who gave me the opportunity to come and work for Hitachi Data Systems in Australia
- Steve Lockrey as well as the colleagues of the HDS APAC support organisation.
- Steve Guendert who taught me more about Ficon than I actually wanted to know. :-)
- The awesome folk at Brocade who have supported me through thick and thin since I installed my first FC switch in 1998.
- All customers and partners whom I've had interesting discussions with to solve problems and provide solutions
- Red Hat and SUSE support folk who are brilliant in diagnosing the most complex of storage related, amongst others, Linux problems.

And last, but certainly not least, my wife Ingrid Simon who has stood by me in good and bad times and has taken my peculiarities for granted (Mostly :-)) . Thank you very much my dear. I love you to bits.

To my kids Aaron, Luna and Cleo, take every opportunity that enriches your lives and enjoy the tasks that life brings you. You three certainly enriched my life, thank you for that.

Limits

When writing a book about technology where a large portion pertains the interaction between software and hardware you can imagine that when availability of equipment is either restricted or non-existent it is hard to show or explain what you mean. It's a bit like trying to describe one of the artworks of the Dutch Master painters. You should really stand in front of it to be able to understand and comprehend its beauty. In this book I will refer to information that was at my disposal at the time of writing. I did rely on a fair bit of virtualisation as well as information I was able to gather and compile over the years. The book does not contain any information that linked to any customer information from any of my previous or current employer(s). The book also does not contain any content that could be perceived as proprietary, trademarked, copyrighted or any other sort that could fall under a protected intellectual property. Images have either been created by myself with PlantUML, have been provided with a CC BY/BY-SA license or are in the public domain.

The book itself is a living `thing` and I plan to update, expand, correct and enhance where possible on a regular basis. If any vendor is willing to sponsor hardware and technical information that could help system administrators, system designers, storage architects, support people and anyone else interested in troubleshooting Linux storage technologies, please contact me via my website https://erwinvanlonden.net.

Prerequisites

There is some knowledge required of storage and storage infrastructures so knowing what a storage array, fibre-channel switch, lun and disks are is assumed. Also some familiarity of the linux command-line interface is required as most tools for troubleshooting purposes are cli based.

Chapter 1

Linux Troubleshooting

Sorry what? This is like mentioning DNA profiling, researching particle physics or mapping the universe. The Linux ecosystem is so diverse and there are so many brands, flavours, colours etc. that it is impossible to touch on each and every single one. From an enterprise perspective there are basically two major players Redhat and SUSE. Yes, yes, I know there is also OEL, Ubuntu, Debian etc etc but I'll cut off at the first one Redhat or a derivative Centos as well as SUSE. Reason is that the underlying technology is not that different between each of them and the methodology remains the same. Thing that may differ are naming conventions, file locations and some tools that may be present in one distro and not the other. I'll leave it up the reader to determine which tools and methods are applicable to his/her environment.

As the above already shows it would be impossible to write a book that covers everything related to storage in a Linux environment. Mainly if that would be the goal you would have to pick a point in time on a Linux kernel version related to one or two distributions and go with that. The moment you've crossed the "T"'s and dotted the "i"'s not only would you be 5 years down the track but the book would be obsolete from the moment you press "Publish". Therefore the goal of this book is to give you some tips, tricks and guidelines to help you troubleshoot issues on a day-to-day basis. When things become really complex I would advise to engage with your distribution vendor for further assistance.

1.1 TL;DR

Troubleshooting a storage environment is not something you'll be able to take on lightly. There are vast amounts of areas were issues may pop up. If you do not have a good knowledge of what storing and retrieving data entails in various infrastructures you'll soon experience you've ended up in a maze. A solid knowledge of these environments and protocols will

not only significantly reduce the chances of design and architecture issues but also expedite resolution time in case things go wrong. The TL;DR part is basically saying there is no shortcut in this. Storage systems can be very complex and 50 years of systems design and protocol development means there is a large ecosystem of software, hardware, protocols and procedures to acquire knowledge on and bring to the table. It is up to architects, designers, operators and systems-administrators to ensure that they obtain that knowledge as much as is required so solid environments will be implemented and operated for business purposes.

1.2 Performance

When it comes to performance I will only touch on a few things very lightly. If you want a deep-dive into the art of performance analysis I can recommend the library of Brendan Gregg over here http://www.brendangregg.com/linuxperf.html

He has a vast library of tools, articles, drawings etc. to guide you through the maze of performance troubleshooting. Not only storage related I must warn you.

1.3 The IO stack

In this book I've use a top-down approach from application to hard-disk and the various layers. In order to visualise the components of the storage IO stack in Linux, Werner Fisher from Thomas Krenn [1] consulting in Germany has created a nice overview. Even though the image references a Linux 4.10 kernel the overall build-up does not change drastically. Certain options and modules may be added or removed in later kernel versions.

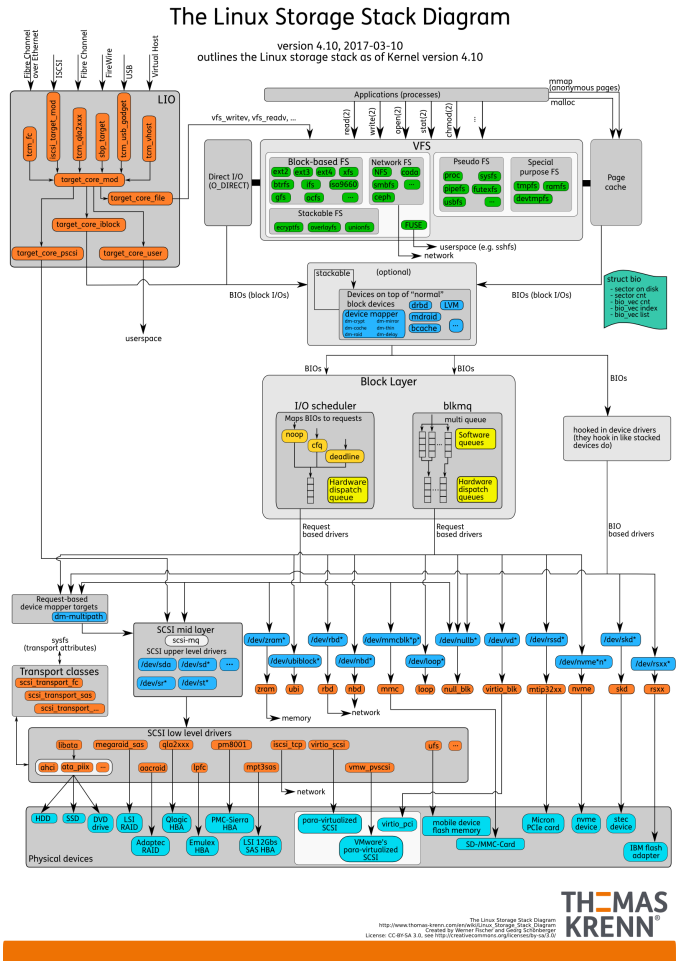


Figure 1.1: Linux Storage Stack

1.4 Applications

Executing IO related commands are application driven. The application has certain characteristics which will affect IO behaviour. A database is most often very read intensive with small request sizes whereas video editing and rendering applications deal with large IO sizes in a relatively even read/write intensive ratio. Large scale IoT environments collect massive amounts of data which get injected into large linear databases which may or may not be time-series related. Example are Apache Spark and Kafka. On a smaller scale server applications like email and file-servers may operate with protocols like SMB and NFS which

also have their own characteristics. Understanding these is imperative not only in designing storage architectures but also the ability to effectively troubleshoot issues.

Info you need :

- IO size usage - When the application executes a request what is the IO size in which it does that.
- Read/Write ratio - This is imperative to know in troubleshooting as it will determine SAN traffic directions and array behaviour massively.
- Data placement - basically meaning determining where certain sections of the application will store its data and how is this retrieved again. Mechanisms like caching, utilisation of temp-space, direct IO capabilities etc. can have a great influence on performance and overall application behaviour.

1.5 Filesystem

Here is another can of worms on the Linux platform. There are over 20 filesystems currently available in the linux kernel from adfs via ext and reiserfs to xfs. Each of them have special characteristics and can impact IO behaviour. Profiling these based on your application and or business requirements is important as moving/migrating later on may be a hard task. Filesystems also have a huge influence on arrays when it comes to provisioning. Some filesystems are very friendly to thin provisioned volumes and others are not. Does that make the first better than the second? Not necessarily, again it depends on you requirements.

From a performance perspective it all depends on how effective the filesystem algorithms handle updates and requests. Some benchmarking on your application and infrastructure may be required to find out what best suits your needs.

Info you need:

- What filesystems are supported by my application
- How is the filesystem layout (is important for index allocation and thin provisioning.) This can be based on btree structures, head of disk indexes, sector based indexes and a few more.
- Journals being used or not.
- Extended functions like snapshotting etc. required and used.

The most used filesystems for block devices are ext4 and xfs. From a network side you'll find NFS and SMB as well as object based systems like Ceph or cluster filesystems like GlusterFS. Others like btrfs, zfs (for linux), vfat etc may sometimes be seen. For the sake of brevity I'll limit the troubleshooting to the block based systems.

1.6 Encryption

Encryption may been seen on many levels from application to the hardware layer and even special appliances integrated in secure PKI based infrastructures. You can get a PhD on this alone so I'll only touch on this briefly.

Info you need:

- Is the disk encrypted or not?
- Is the encryption engine offloaded to a special hardware device?
- Which solutions are involved in the encryption and how do they interact on the IO-level

1.7 Partitions and Volume Managers

Partitioning and Volume managers do not directly interfere with the IO requests themselves. There are merely there to provide a logical abstraction layer for addressable space. They therefore do not have a direct impact on the individual requests however when designed or configured incorrectly they can cause the IO path to become a severe bottleneck.

When SAN infrastructures are used and large amounts of luns are presented it is fairly easy to make a small mistake with large consequences. Improper configuration can at some stage become a major burden and can take numerous sleepless nights to correct.

This chapter is fairly detailed with numerous examples as the vast number of functions and features may have a significant impact on installation, operations and troubleshooting.

Info you need:

- Detailed mapping of physical disks to logical volumes and vice-versa.
- pvs, vgs, lvs and/or output of other lvm commands.
- 3rd party volume managers installed.

1.8 RAID

The Linux OS has a native software based raid functionality build into the kernel. The md-raid module allows for mirroring or striping data across disks. This is one of the layers that can have the most impact when designed incorrectly. In test and development environments, which have some sort of permanent nature, testing of primarily OS and application functionality shouldn't be much of a problem. When using this in a production environment you need to be aware that a significant overhead will occur in splitting the IO's (in case of RAID 1 mirroring) or RAID 3/5 (with a heavy computational load of XOR calculations) which require segments of the data to be striped over multiple physical disks. In almost all cases it is better to have a hardware based array option or external arrays which have purpose build ASICs and FPGAs for this.

Info you need:

- mdadm output if software raid is used.
- RAID hardware configuration if applicable.
- External storage setup if applicable.

1.9 Block devices

The lowest addressable entity of a range of storage space is represented as a block or tape device. These are shown in Linux under `/dev/sdx`, `/dev/hdx`, `/dev/stx` etc. The block device is more or less a logical representation of a hardware device whether this is build into the system or presented from an external storage array or tape device. This is also the level were most IO errors are reported on. Any hardware or performance issue will be presented out of this layer. It depends on the configuration of the logging facility where this will then end up but most often you'll see this in `"/var/log/messages"` or `dmesg` output. The block devices are instantiated upon discovery of devices via the respective device drivers. As soon as an array has created a logical unit (LUN in case of SCSI) or NS (NameSpace in case of NVMe) the device driver of that card will instruct the block layer to create an addressable device after which a discovery/inquiry process takes place of what that device then is. Based on that it is the `udev` process which will provide it an addressable name (`/dev/xxxx`). It is then available for use. The design and architecture of the system will then determine how it will be used with the aforementioned volume managers, file-systems etc...

The block devices in a Linux system do need to have the ability to interact with the operating system, they obviously need to be able to store and retrieve data, respond to commands like rewinding a tape or any other command applicable for that particular device. In order to be able to do this we use communications protocols. You may have heard of ESCON, HIPPI, IPI, SBCCS (yeah yea I know mainframers would be more across this....) but in the Open Systems market we see primarily two, SCSI (Small Computer Systems Interface) and SATA. "_Huhh, what about Fibre Channel? And NVMe?". Well FC is not a device communications protocol but more a transport protocol which has aligned characteristics to channel protocols like SCSI and SBCCS.(We'll get to transport layer later on) The bulk of installations at the time of this writing is based on the SCSI (Small Computer Systems Interface) protocol but NVMe is making great progress and is now seen in almost every system whether it being your desktop or a large scale SAN infrastructure.

Info you need:

- In a troubleshooting scenario a list of affected devices.

1.10 MultiPath IO

In an environment where devices can be reached via multiple entry-points a layer of software called multipathing is used. If the discovery and addressing of the individual devices has been done the MPIO software can query the inventory parameters of the devices. If it determines that two or more of the devices in the device-tree are actually one and the same (we'll get to that later) it can aggregate these in a virtual device and IO's directed to this virtual device can/will be dispersed over each of these paths. It depends on the capabilities of the attached storage device what its capabilities are, how it is connected and how the MPIO software is configured. The MPIO layer is the best layer suited for problem prevention but it has its limitations. (See the article over at my site [2] why MPIO sometimes doesn't seem to work.)

Info you need:

- Diagram of the infrastructure
- Overview of presented volumes over which paths.
- Capabilities of the external storage device

1.11 Schedulers

There are two types of schedulers in the Linux environment. Process and storage schedulers. This book focusses on the storage side.

The decision which IO request gets handled by the protocol subsystem is determined by the IO scheduler. In Linux there are a few available depending on which flavour distribution you have or which ones are compiled into the kernel. The schedulers are split into two segments: multi-queue and non-multi-queue. Especially in the NVMe (see below) arena a multi-queue scheduler should be used. If the system has an older kernel it is advised to upgrade. In the latest kernels (~5.2 and up) the non-multi-queue schedulers are no longer supported or advised. The schedulers themselves are not often the cause of storage issues where hard IO errors are observed. They do sometimes come up in performance related problems where a shift in IO profiles from the applications may have an adverse effect in how the scheduler in play at that moment may not be the most effective. Ongoing tests with IO profiling and replay of IO patterns/workloads in benchmark tools will show which scheduler may be best suitable.

Info you need:

- Overview of schedulers used by affected devices.
- Can be obtained via "`cat /sys/class/block/<block-dev>/queue/scheduler`". Output will show something like :

```
[root@server queue]# cat scheduler
[none] mq-deadline kyber bfq
```

1.12 Storage Protocols

Protocols in the storage world can be split basically in two parts, the channel protocol and transport protocol. With channel protocols the command and control of peripherals are known and directed by an initiator. It is the initiator that keeps track of everything that happens to and from the devices that are under its control.

It is aware of the state of the device itself, can inquire on the state and can act on changes of these devices as wel as the responsibility of providing an address schema how and where the devices can be reached. The initiator is also responsible for storing and retrieving the data as requested by the upper levels. View it as the central control room of storage operations.

The transport protocol is more responsible for getting the command and control instructions as well as the data between the initiators and targets. It may not have any knowledge of the upper layer protocol.

1.12.1 Channel

1.12.1.1 SCSI

SCSI is the most prevalent protocol as mentioned before. It has a huge ecosystem build up over +- 40 years. Its wide range of equipment that uses it means that it is also relatively complex and you may take a while to grasp the concepts. The protocol was not designed with large scale storage infrastructure in mind. The acronym already shows that: "*Small Computer Systems Interface*". The architecture of the protocol allowed for huge flexibility. When SCSI was designed it also was tightly coupled with the hardware. The entire hardware signalling and data-transfer over many sequencing phases more or less hampered the development somewhat. As soon the command set was uncoupled (disassociated) from the hardware the true power of SCSI was unleashed. Large scale storage infrastructures as we know them today are the result of that.

Info you need:

- A series of commands that outline the devices and connection tree
- `lspci` - to see which adapters are installed.
- `lsblk` - overview of block devices and parameters
- or alternatively `lsscsi`
- Each command has a series of parameters to adjust the query and output

1.12.1.2 NVMe

Over the last few years the NVMe protocol has gained a signification traction. The development of NVMe grew out of the PCI-Express world which wanted to use the massive increase in data-transfer and IO rate of flash drives. It has significant benefits when it comes to addressing schema, queuing and performance over SCSI. Since around the middle of the

2010 decade, the NVMe over Fabrics allows the protocol part of NMVe to be transported over Fibre-Channel, Ethernet, RoCE and other fabrics. In a Fibre-Channel environments this was relatively easy to do on the FC4 layer where a extension of the protocol mapping was added. Benefit is, the underlying FC infrastructure remains 100% the same. This is an important part of the troubleshooting approach in case things do not work as expected.

Info you need:

- Output of various *nvme* or *nvmetcli* commands

1.12.2 Transport

The transport protocol is basically responsible for getting command/control information and data from A to B and back. It does not interfere with data or device-handling. Transport protocols can be somewhat split into two distinct categories

- Network oriented
- Channel oriented

Network oriented protocols do not have a distinct knowledge of the data that is being transported. For example a network switch does not inherently take into account specific http characteristics in order to make decisions on switching packets. The configuration is mostly set on the datalink or network layer (L2 and L3 of the ISO model). Specific instructions need to be set on these levels to influence data-transport behaviour.

Channel based protocols have inherent knowledge of the data that is being sent. The frame headers contain specific information that allows data flows determine the data type (Control, Data etc.) as well as align with communications characteristics. One example of this is that FibreChannel is designed to be 100% full-duplex however it has provisions to adhere to the asynchronous behaviour of SCSI for example.

One or the other may have inherent limitations. As mentioned a network oriented protocol does not take into account a channel based requirement. Protocol dependant timers will differ between each of them. Additional layers like for instance iSCSI need to overcome that and allow the SCSI protocol to be transported over TCP/IP. Similarly some distance limitations that may incur on the Fibre-Channel protocol may need to be overcome by using FCIP [1]. Additional precautions need to be taken to prevent the storage protocol be a victim of the characteristics of the lower layer TCP and IP ones. When Wide Area Networks (WAN) are in play even more planning and configurations need to be implemented to allow a reliable communication of the storage based protocol and therefore the data.

Whichever transport option is chosen it is of the utmost importance that a thorough understanding is obtained of the requirements and have this mapped onto the available infrastructures in line with the business targets that have been set. What I mean by this is that a

[1] Fibre Channel over IP

lot of thought needs to be put in designing the transport layer in order to prevent technical limitations having a negative influence on business requirements.

Info you need:

- Up-to-date diagram outlining all characteristics of the transport layers between every end-point.
- Workload profiling mechanisms to enable scale-up, scale-out or scale-down planning.
- Planning document of recovery in failure scenario's.

1.13 Storage

The storage landscape diversity is enormous. Especially when it comes to media, vendors, models, functions, features etc. Usage of technologies like snapshots, replication, long distance connections make this layer especially complex. There are numerous things that can go wrong on the storage side especially from an operations side. Executing commands on storage array that inadvertently remove volumes from being presented or replication direction accidentally going the reverse way are classic examples.

Each of the above subjects will be touched upon in subsequent chapters.

Chapter 2

Applications

Troubleshooting storage issues on Linux from an application perspective is imperative to start with together with the Operating Systems logging facilities. Issues seen related to performance or erroneous IO behaviour can, and most often will, cause outages. Application configuration and behaviour will have a significant impact on storage operations. A well designed and configured application will not only benefit from functioning without issues but will also prevent negative impact on other parts of the storage infrastructure.

Starting troubleshooting from an application viewpoint will most likely give you a feeling what the issue may be and where it may originate from. This will however not always stand true, especially when hosts are connected to a shared storage infrastructure. We dive further into that in the chapters that follow.

2.1 Type

In this book I'll touch on a few applications types

1. Databases
2. Cluster systems
3. File servers

2.2 Databases

This book primarily focusses on the commercially available ones like Oracle and Sybase (now part of SAP) and freely available ones like MySQL and Postgres.

The relational databases have a structured layout. That means the content of the databases is predefined so the tables, column and records have a known content type and its relation

to others is known. This allows for specific layout of the database itself and targeted storage provisioning ensures that an optimal behaviour can be achieved.

General rules are:

1. Split transaction logs and databases
2. Store each on appropriate volumes.
3. Provision volumes in line with the IO behaviour of the specific database and transaction logs. In general the transaction logs are primarily write intensive whereas the databases themselves have often a 80/20 read/write ratio. (Depending on the applications that use that database)
4. Determine if special storage management layers are in play. Examples are Oracle ASM and Veritas Storage Foundation.

Especially the latter (#4) will significantly influence the behaviour of databases in relation to storage operations. In many cases the storage infrastructure simply presents a large amount of volumes with different performance and availability characteristics after which the database administrators (DBA's) configure the layout of the provisioned space to the respective databases. From an operational perspective this may look very beneficial. The administration on the storage infrastructure side is kept to a minimum and the administrators can use the provided space to their liking. The major drawback is that the DBA's in general do not have an good overview of the storage infrastructure nor is there control on the access characteristics imposed on, or requested from, the storage infrastructure. The consequences of this is that in some cases an incorrect configuration on the DBA side can have significant negative impact on that underlying infrastructure. Mistakes whereby transaction logs are put on striped RAID configurations or on storage pools where array cache limitations are imposed can cause sever performance and even availability issues. In general every configuration from a DBA perspective should be validated and approved by the storage administrators. They need to ensure that appropriate processes are in place to monitor the infrastructure and adjust parameters as appropriate.

Many databases have separate database structures in memory and on disk. Be aware that these significantly influence database behaviour and thus IO activity. It is important to pay special attention to these parameters in order to optimise the applications caching and storage side.

2.2.1 Oracle

One of the most prevalent database systems deployed is Oracle. Oracle is able to utilize its own storage management stack called ASM. The ASM suite contains in essence three components. ASM (Automated Storage Management), ACFS (ASM Cluster File System) and ADVM (ASM Dynamic Volume Manager). ASM can work with Linux native options like MPIO but utilizing ASMLIB or it can replace them entirely such as in the case of the ASM volume manager replacing the Linux LVM variant. It is up to the business to decide whether any of these variants are used.

The ASM suite is optional. There is no technical requirement to use ASM on Linux as the native tools work very well and can be optimized for database operations. The device mapper, LVM , available file-systems etc provide ample options to configure a reliable system.

Most issues related to Oracle databases and ASM are caused by a design issue or misconfiguration. The error-messages that are logged from an Oracle database instance related to Linux storage are not specifically identified as a separate problem category but will be logged depending on which process and component is observing a problem. The ones that do have their own category are the ASM error messages. Disk read write IO messages are primarily logged under a few ORA-xxxx messages. These may differ on certain versions of the database.

Examples are:

```
ORA-00200: control file could not be created
Cause: It was not possible to create the control file.
Action: Check that there is sufficient disk space and no conflicts in filenames
and try to create the control file again.
```

or more specific to read and write errors are

```
ORA-00204: error in reading (block string, # blocks string) of control file
Cause: A disk I/O failure was detected on reading the control file.
Action: Check if the disk is online, if it is not, bring it online and try a warm
start again.
If it is online, then you need to recover the disk.
```

and

```
ORA-00206: error in writing (block string, # blocks string) of control file
Cause: A disk I/O failure was detected on writing the control file.
Action: Check if the disk is online, if it is not, bring it online.
```

The two messages above will most likely also have a OS based counterpart error logged in the `/var/log/messages` file that derives out of the protocol subsystem, volume manager or filesystem. That message will also have more detailed info which underlying volume it entails. When we come to the OS side we'll have a more closer look into this.

Oracle database issues mainly touch on performance problems. Delay in read/write behaviour are most often due to poor design or selection of equipment not up for the task. Oracle lists disk layout as #5 on their top 10 list of performance problems.

Parameters that should be evaluated per installation related to IO behaviour are **DB_BLOCK_SIZE** and **FAST_START_MTTR_TARGET**. The first one configures the size of the data-segment stored in cache as well as the IO size sent to disk. In general this is set to 8192 or smaller for mostly transaction based systems and higher for data-warehousing environments. The `FAST_START_MTTR_TARGET` is a timer that triggers

redo-log checkpoint. This ensures the redo-log does not grow out of proportion and causes disk space issue or excessive write delays or, in case of a very small value, an increase in the number of database writes may incur additional delays. Many additional parameter have an influence on overall performance for Oracle databases.

The Oracle performance tuning guide lists the configuration parameters which have an impact on IO behaviour. The below are a few of them.

Table 2.1: Oracle Database IO parameters

Parameter	Description
DB_BLOCK_SIZE	The size of single-block I/O requests. This parameter is also used in combination with multiblock parameters to determine multiblock I/O request size.
OS block size	Determines I/O size for redo log and archive log operations.
Maximum OS I/O size	Places an upper bound on the size of a single I/O request.
DB_FILE_MULTIBLOCK_READ_COUNT	The maximum I/O size for full table scans is computed by multiplying this parameter with DB_BLOCK_SIZE. (the upper value is subject to operating system limits). If this value is not set explicitly (or is set to 0), the default value corresponds to the maximum I/O size that can be efficiently performed and is platform-dependent.
SORT_AREA_SIZE	Determines I/O sizes and concurrency for sort operations.
HASH_AREA_SIZE	Determines the I/O size for hash operations.

Oracle has separate chapters on storage profiling and IO configuration in the performance section of the documentation. [3]

2.2.2 Sybase (SAP ASE)

The SAP ASE database is the underlying databases for many of the SAP ERP solutions. This is not the same as HANA. The configuration of the databases are similar as for every database. IO issues are often more related to performance which leads back to design/architecture and the capabilities of the underlying hardware. Similar as with the Oracle database placement of the transaction logs and tables on the proper volumes make a world of difference. Unfortunately SAP did not provide me with access to their portals so I was

not able to extend the troubleshooting section with their information any further. I will update the book if/when that changes]

2.2.3 Microsoft SQL server

SQL server from Microsoft is supported on Linux since version 2017. It runs on RedHat, SUSE, Ubuntu and even deploying it on Docker is now a valid configuration. Extensive documentation can be found on Microsoft's SQL Linux pages. [4].

There are a few gotchas that are mentioned in the various release notes that may change over time. The usual dependencies on CPU, memory and diskspace are followed by restrictions in filesystem types. (Only `ext4` and `xfs` are supported at this stage) That means that "exotic" file-systems like `btrfs` and `zfs` are not an option (yet).

2.2.4 MySQL

A somewhat less "enterprisy" level database, albeit immensely popular, is MySQL. Originally an open source project which got commercialised by SUN Microsystems and subsequently became part of the Oracle portfolio. The open source variant is MariaDB which can be freely used and deployed. From a storage configuration perspective it does not deviate much from its enterprise siblings. MySQL knows a concept of "Storage Engines". These engines provide different database storage characteristics and can interface with external storage providers like Casandra and S3. The different characteristics of these storage engines vary to different functions like optimisation for certain workloads or storing data for space efficiency. It is up to the DBA's to select the correct engine. (See Choosing the right storage engine) [5] or the MySQL storage engine page. [6]). Depending on the engine certain underlying disk architectures may need adjustment. An example is if the `Archive` engine is chosen there would not be a need for additional compression options to be enabled in SAN networks or storage arrays. Severe performance problems may occur if a particular engine is chosen and incorrect parameters in the infrastructure are configured.

2.2.5 Reliability and Redundancy

Build in reliability and redundancy features are available on multiple levels. Replay and undo logs, log-shipping, cluster environments and replication options are well developed in all databases. On the data level there are hashing functions and other integrity checks that verify if the data that was entered is properly stored. Replication on the storage layer requires active integration between the applications and arrays to ensure consistency between the application and storage entities. Databases need to be "quiesced" and subsequently the outstanding transaction been committed. This will result in a consistent state between the primary and secondary volumes when snapshots or other sorts of point in time images are generated.

Most backup and/or other business-continuity software have integrated options to care of

this. If manual commands or scripting tools are utilised be assured the proper database state has been reached and no update sessions are still active before commencing with activities on the underlying storage layer.

2.3 Cluster systems

Cluster computing comes in a wide variety of options. On an application level these are most often managed by internal replication and locking scenarios between two or more entities. The cluster functionality in applications take care of the distribution of data and determines which instance has read and write or update permissions. These permissions can be active/passive or active/active between the nodes where the application is running. The granularity is depending on application capabilities and configuration. They can be as granular a single records in a table although performance impact can be expected.
Other storage options like Apache Spark and Hadoop are also an option. That is beyond the scope of this book.

2.3.1 Operating System clusters

2.3.1.1 BeoWulf

For storage troubleshooting purposes on a Linux system a BeoWulf cluster will not have special operational procedures. A BeoWulf cluster is more used for parallel processing of tasks feeding back the information to the requestor. It does not use a distributed storage model with dispersed data-storage. It can utilise external storage facilities such as Hadoop.

2.3.1.2 Apache

Many software bundles that provide a large scale distributed storage environment are listed and licensed under the Apache Software Foundation. A few of them are Spark, Hadoop Hive, Kafka and more. Each of these have their own procedures and guideline for configuring and troubleshooting storage related issues. These are not necessarily Linux related but more from an application perspective. Refer to the Apache website for more information.

Chapter 3

Filesystems

The filesystem layer is one of the most important and complex layers in the IO stack. Not only are there a plethora to chose from but the internals can massively differ between each of them.
The filesystem is the only layer in the IO stack which has an active knowledge of the data that is stored. It knows which files and directories reside on disk and how these are structured. It performs allocation and de-allocation of space, calculates optimum caching algorithms, provides permission options, snapshot capabilities, quota properties, journalling etc. Based on the requirements of the application and/or business one filesystem may be more suited than the other. If for example snapshot functions are required then `btrfs` or `zfs` may be more suited than `ext4`. If high performance for large files are needed maybe have a look at `xfs`.

3.1 VFS

The Linux filesystem architecture is build upon a layer called `VFS` [7]. The Virtual File System provides an abstraction layer upon which userspace programs can interact and filesystems like `EXT4` and `XFS` plug into. The handling of read or write calls or allocation of inodes is therefore transparent to any application that wants to execute IO's and they do not need to contain code specifically for each individual filesystem. It would be very cumbersome if application would need to have different code to be able to read a file depending on which underlying filesystem it would reside. From a userspace perspective you also don't really need to worry about it as it is *just there* and the tools that are out there to interact with vfs all use those systemcalls.

As an example is the `mount` command. This command is transparent across every file system that you may use and it is only the different attributes, functions and features of the

underlying filesystem that adjust their behaviour.

When looking back to the diagram of Thomas Krenn we see the VFS filesystem layer is roughly represented as follows

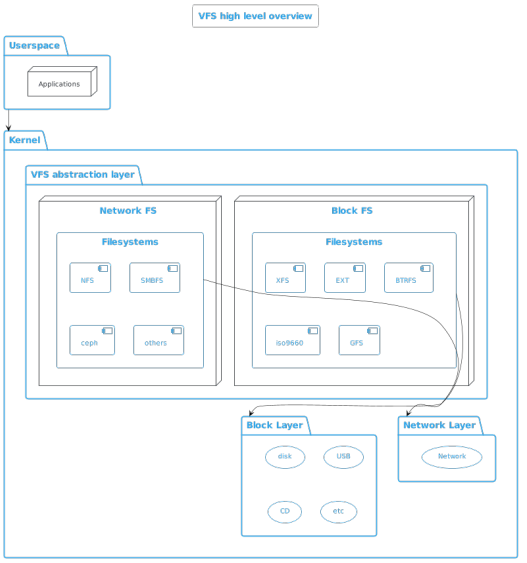


Figure 3.1: VFS

*fig:vfs basically shows that the Virtual File System is providing the abstraction for the filesystems that can be used.

3.2 Disk capacity

Disk capacity as per ISU standards is done on a per 10^x basis. Basically meaning that it is expressed as a decimal power. Operating systems however calculate in binary powers like 2^x.

If you start discussing storage capacity but use the incorrect notation it means that a "kilobyte" may accidentally mean "kibibyte" with the result of have a 24 byte discrepancy per entity.

Table 3.1: Size notation.

Notation	Size in binary	Size in decimal
Kilo	$2^10 = 1024$	$10^3 = 1.000$
Mega	$1024^2 = 1048576$	$10^6 = 1.000.000$
Giga	$1024^3 = 1073741824$	$10^9 = 1.000.000.000$
Terra	$1024^4 = 1.099.511.627.776$	$10^12 = 1.000.000.000.000$
etc etc...		

That obviously will cause some discrepancy between the capacity you require to store "stuff" in relation to what the disk has on a label. The decimal notation is expressed in KB, MB,GB,TB etc whilst the binary notation shows KiB,MiB,GiB and TiB. For more information see the definitions on Wikipedia. Byte [8] and International System Units [9].

When you buy a 2 terabyte disk you will basically get a set of platters or chips with a total capacity of $10^12 * 2 = 2000000000000$ bytes. When presented to the operating system, who needs to format it, you will see an addressable size of $(10^{12)/(1024}4)*2~1.818 terra-bytes.

Depending on your needs you may want to select a filesystem based on maximum file-system or file size.

Table 3.2: File System Sizes.

FS	FS Size	File size
Btrfs	16 EiB	16 EiB
Ext3	16 TiB	2 TiB
Ext4	1 EiB	16TiB
XFS	8EiB	8 EiB

One little tip I need to mention is the `-x` parameter on the `du` command. You've most likely seen that when trying to determine the usage of a filesystem with many volumes mounted on various locations in that directory tree, it takes A: long time to parse them and B: it accumulates everything under that directory tree. This is most often not what you want as the space allocation is set on a volume basis and not a directory tree with different mountpoints. The `-x` parameter will only show the disk usage for the volume where that directory is located and will exclude all other mountpoints in that tree.

3.3 Layout

The filesystems in Linux, or filesystems in general, are a in essence a large reference index to data stored on disk. This data is stored in blocks or series of contiguous blocks called

extents. If a file gets split, for example it gets fragmented, it will be stored in other block-groups/extents. The information of these files and directories are stored in so called inodes. Layout of all these are then referenced in one or more allocation or block groups depending on the terminology used for that respective filesystem. The way this is done differs per filesystem.

The size of the underlying disks (i.e. the number of physical blocks presented by the hardware, volume or partition) as well as the architecture of the system (32 or 64 bit) determine the layout of the filesystem which subsequently provides the storage space available to applications. Filesystems like `btrfs`, `xfs` and `zfs` use `btree` structures which allows advanced functions like snapshotting and automatic corruption detection and recovery in the filesystem itself. Other filesystems like ext2/ext3/ext4 user block groups and group descriptors. Each of them use, so called, `inodes` to map metadata to the actual data. "*Data*" being a loose description of a file, directory, or device. These `inode` entries are most often stored at the beginning of a blockgroup although the actual data may be stored on a different block-group. This may/will cause fragmentation over time, and may result in additional seek times on rotational disks and additional requests on both rotational and ssd devices. Be aware that most filesystems have many tricks up their sleeve to prevent these things from happening.

Example of a file-entry in filesystem technical detail:

```
[admin@server ~]$ stat test1.rnd
  File: test1.rnd
  Size: 409600      Blocks: 896        IO Block: 4096   regular file
Device: fd02h/64770d  Inode: 8329        Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/ admin)   Gid: ( 1000/ admin)
Context: unconfined_u:object_r:user_home_t:s0
Access: 2020-05-21 15:34:00.256607506 +1000
Modify: 2020-05-21 15:34:00.259607506 +1000
Change: 2020-05-21 15:34:00.259607506 +1000
 Birth: -
```

The above show the underlying filesystem information in relation to the file `test1.rnd`.

3.4 Mount options

The various filesystems have a generic part which is the same for all and has primarily options on how the filesystems are presented to the host for use by the applications. This relates mostly to the mount location, behaviour of handling the inodes, access/modification times etc. The filesystem specific options do target the internal functionality of the filesystem in play.

As these options can sometimes vary across distributions and how they are compiled it's advised to look at the man pages for them. These are very well documented and maintained.

Table 3.3: File System Man Pages.

File System	Man Page
btrfs	btrfs(5)
cifs	mount.cifs(8)
ext2, ext3, ext4	ext4(5)
fuse	fuse(8)
nfs	nfs(5)
tmpfs	tmpfs(5)
xfs	xfs(5)

3.5 Selecting and creating a filesystem

So which filesystem to choose. It is somewhat analogous to the question "How long is a piece of string". The criteria of which filesystem to use needs to be defined by the demands of the application in both features, functions and performance as well as the operational manageability of the IO stack of the system. Is your organisation using `LVM` as the foundation of the volume management architecture there may not be much benefit of looking into `btrfs` or `zfs`. When you're looking for alternatives of `LVM` these obviously are serious contenders. The majority of Linux distributions come with either the `btrfs` or `ext4` as a default. Both filesystems provide excellent performance, features, functions and reliability. Ext4 is basically a fork of `ext3` with enhanced functions and features. It has journalling enabled by default but also allows this to be disabled via tunables. SUSE comes with `btrfs` whilst RedHat ships with `ext4` by default.

Depending on requirements of the applications and system resources a general rule-of-thumb is that `ext4` is a very good general purpose one, `btrfs` has many functions and features but is somewhat more resource hungry and `xfs` is ideal for applications that store very large files as well as performing better in large multithreaded application environments. You will see the latter most often used in the media landscape where large video files are stored.

There is not really a silver bullet approach on which one to select and you would need to benchmark the different filesystems against your application. Ensure that all other parameters such as hardware and infrastructure are the same when you test this. If you change the landscape you will get incomparable results which in essence are useless.

When using any file-system it is pretty important to have a good look at the mount-options that are available as some options may work on one but not the other. Some options also have different use cases or some do something different even whilst having the same name. Study the `mount 8` man page carefully as there are two sections which describe FS dependant and in-dependant options.

The next sections describe processes and diagnostic procedures to create and fix filesystems

in case something goes wrong.

3.5.1 EXT4

The `ext4` filesystem is one of the most used. It's flexibility, backwards compatibility with ext3, stability and reliability have made it a preferred choice for many distributions to ship this as the default filesystem. `ext4` was basically a snapshot copy of `ext3` where additional functions, features were added as well as incorporating supported for larger systems, architectures etc. Many of these have later been back-ported to `ext3`. A very informative video of how `ext4` came about can be seen on youtube. See reference [10].

When creating the ext4 filesystem an underlying device is expected to be writeable. (duhhh) The main reason I highlight this is that in a few cases I handled attempts were made to create filesystems on a read-only device. This is not impossible or unthinkable to present a read-only device out of an array. (Obviously this goes for all disks or volumes you want to provision with a filesystem.) This can be a raw device (`/dev/sdx`), a logical volume or a virtual device linked to physical devices such as in case of multipathing or even a combination of those.

The below shows the creation of a ext4 filesystem with a blocksize of 2KB onto a multipath device.

```
[admin@server ~]$ sudo mkfs -t ext4 -b 2048 /dev/mapper/mpathf
mke2fs 1.44.6 (5-Mar-2019)
Discarding device blocks: done
Creating filesystem with 5242880 2k blocks and 655360 inodes
Filesystem UUID: c73c6616-6ce3-4663-9cb0-de7596410de2
Superblock backups stored on blocks:
  16384, 49152, 81920, 114688, 147456, 409600, 442368, 802816, 1327104,
  2048000, 3981312

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

The same process on a logical device which is made out of two multipath devices (`mpathg and mpath`) combined in a volumegroup `lst`. A logical volume `lvol0` is carved out of this and a `ext4` filesystem is created.

On a volume group:

```
[admin@server ~]$ sudo pvcreate /dev/mapper/mpathg
  Physical volume "/dev/mapper/mpathg" successfully created.
[admin@server ~]$ sudo pvcreate /dev/mapper/mpathh
  Physical volume "/dev/mapper/mpathh" successfully created.
```

```
[admin@server ~]$ sudo pvscan
  <snip>.....
  PV /dev/mapper/mpathg                      lvm2 [10.00 GiB]
  PV /dev/mapper/mpathh                      lvm2 [10.00 GiB]
  Total: 4 [575.67 GiB] / in use: 2 [555.67 GiB] / in no VG: 2 [20.00 GiB]
```

Creating the volumegroup

```
[admin@server ~]$ sudo vgcreate lst /dev/mapper/mpathg  /dev/mapper/mpathh
  Volume group "lst" successfully created
```

```
[admin@server ~]$ sudo vgdisplay lst
  --- Volume group ---
  VG Name               lst
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  4
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               19.99 GiB
  PE Size               4.00 MiB
  Total PE              5118
  Alloc PE / Size       0 / 0.00 GiB
  Free  PE / Size       5118 / 19.99 GiB
  VG UUID               Z8NMyi-Kt1W-20hl-cYk8-JXmK-MW0U-6egfGj
```

Carving a logical volume out of this

```
[admin@server ~]$ sudo lvcreate -L 5G lst
  Logical volume "lvol0" created.
```

Creating the filesystem:

```
[admin@server ~]$ sudo mkfs -t ext4 -b 4096 /dev/lst/lvol0
mke2fs 1.44.6 (5-Mar-2019)
Discarding device blocks: done
Creating filesystem with 1310720 4k blocks and 327680 inodes
Filesystem UUID: 4d93edea-054c-4d76-8bd3-5882bf7e0183
```

```
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912, 819200, 884736

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

As you can see the process itself is fairly straightforward the volumes can now be mounted and used for whatever purpose they need to serve.

3.5.1.1 ext4 data corruption

As ext4 is a journalled filesystem, unless disabled manually, any data that gets created or updated is first written to a journal. The output of the `mke2fs` command shows that the superblock is stored in block 0 and copied across other blocks in the filesystem as per the output above. From there on other parts of the filesystem can be found and tracked. Corruption of the superblock in block 0 will therefore be not of particular concern and can be fairly easily repaired.

A normal superblock at block 0 of the filesystem looks like this:

```
 Group 0: (Blocks 0-32767) csum 0x08ed [ITABLE_ZEROED]
  Primary superblock at 0, Group descriptors at 1-1
  Reserved GDT blocks at 2-640
  Block bitmap at 641 (+641), csum 0xd6e970ca
  Inode bitmap at 657 (+657), csum 0x785a30f5
  Inode table at 673-1184 (+673)
  23901 free blocks, 8182 free inodes, 1 directories, 8178 unused inodes
  Free blocks: 8866-8869, 8871-32767
  Free inodes: 11-8192
```

Artificially corrupting it via `dd if=/dev/zero of=/dev/lst/lvol0 bs=4096 count=1` will result in the following‘

```
[root@server ~]# dumpe2fs /dev/lst/lvol0  | head -100
dumpe2fs 1.44.6 (5-Mar-2019)
dumpe2fs: Bad magic number in super-block while trying to open /dev/lst/lvol0
Couldn't find valid filesystem superblock.
```

By checking the filesystem, even while mounted, you can see that it is not “***clean***”:

```
[root@server mnt]# dumpe2fs -o superblock=32768 /dev/lst/lvol0  | head -100
dumpe2fs 1.44.6 (5-Mar-2019)
Filesystem volume name:   <none>
Last mounted on:          <not available>
```

```
Filesystem UUID:          4d93edea-054c-4d76-8bd3-5882bf7e0183
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index \
                          filetype extent 64bit flex_bg sparse_super \
                          large_file huge_file dir_nlink extra_isize metadata_csum
Filesystem flags:         signed_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              327680
```

When running fsck it knows the volume was not unmounted cleanly due to the corruption of the superblock

```
[root@server mnt]# fsck.ext4 -y -b 32768 /dev/lst/lvol0
e2fsck 1.44.6 (5-Mar-2019)
/dev/lst/lvol0 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
Free blocks count wrong for group #0 (23897, counted=23860).
Fix? yes
<snip>
Padding at end of inode bitmap is not set. Fix? yes

Block bitmap differences: Group 0 block bitmap does not match checksum.
FIXED.

/dev/lst/lvol0: ___** FILE SYSTEM WAS MODIFIED ___**
/dev/lst/lvol0: 184/327680 files (25.0% non-contiguous), 447167/1310720 blocks
```

As the references in the backup blocks are still ok you will end up with a clean filesystem again.

Be aware that files that may be corrupted do not have such a backup mechanism. You will need to ensure that off-system backups are kept or techniques like snapshotting are in place. Be aware that replication on the hardware or volume layer will NOT safeguard you against file or filesystem corruption. If processes or commands result in filesystem corruption on the hardware or volume layer these are just ones and zeros without any further meaning. This means that all bytes, representing valid or corrupt data, will simply be replicated to the

secondary volume.

3.5.1.2 Journal

A journal of a filesystem is basically a sequential log of all updates that have been executed in memory but not yet physically applied to the filesystem. Without it, only a block by block scan of the entire volume may be able to recover a dataset on a disk after a crash as all knowledge o the in-memory changes are unknown after a reboot. If a systems crashes, for whatever reason, the filesystem will be checked due to the fact the so called "dirty bit" is checked. It that bit is set is means it was no cleanly unmounted and therefor needs to run a check to see if there are outstanding actions in the journal that need to be applied before it can provide a clean state to the operating system and applications.
The journal in an `ext3` and `ext4` filesystem is optional and can be disabled. this may improve performance on write heavy applications but obviously this comes at the expense of recoverability of that particular filesystem instance. The reason you still may want to disable the journalling function is if you have a very distributed application environment which already has build in data replication/distribution. As mentioned in the introduction chapter systems like Hadoop, Spark and many others provide this and thus may obtain some performance benefit when journalling is disabled.

EXT4 is a very reliable filesystem and, as mentioned before, is a very good choice for a multitude of applications.

3.5.1.3 Behaviour on errors

The `EXT3/4` filesystem has a, so called, on-error flag which allows you to set the behaviour of the filesystem when it encounters errors. These errors can be anything from journal discrepancies to IO errors to corruption. There are three action items that `EXT` has.

- continue
- remount-ro
- panic

The first option only logs errors in the kernel eventlog, the second option will remount the file-system Read-Only basically disallowing all writes and the third will panic the kernel forcing a panic-dump and restart of the system. The latter is mostly useful in a development environment where firmware and drivers are being installed and tested.

Most distribution use the `continue` setting as default when creating the file-systems. You can alter that with the `-e` parameter when creating the file-system or use `tune2fs -e <continue|remount-ro|panic>` to dynamically change that. Be aware that applications such as databases might also change this flag at the time they are instructed to use that volume.

All errors will result in the state of the file-system be marked as "dirty" which will cause `EXT` to start an `fsck` again it and verify the integrity upon system restart or remount. The invocation parameters determine how `fsck` will be executed.

```
[ server % ~ ] sudo tune2fs -l /dev/mapper/vg_server-lv_home
tune2fs 1.45.5 (07-Jan-2020)
Filesystem volume name:   <none>
Last mounted on:          /run/media/erwin/1701acf8-024b-4827-9bc1-68a71018451a
<snip>
Filesystem state:         clean
Errors behavior:          Continue
<snip>
```

There are a fair few options available. The invocation scripts and default settings may differ per distribution so please check the documentation and man-pages.

3.5.2 XFS

First things first. The `XFS` filesystem does not originate out of the Linux ecosystem It was designed by SGI [11] in 1993. A company who's assets ultimately ended up with Hewlett Packard in 2016. SGI built its systems for graphical processing and graphical printing machines. One thing all data has in common when it comes to graphics is that their file-sizes are huge. It is not uncommon that individual files run in the hundreds of gigabytes and that massive back-end storage systems are responsible for serving huge IO requirements. It is therefore that the XFS filesystem has a background of performing very well under these circumstances. Its codebase caters for massive size and scale of files and parallel execution of threads against the same file(s).

3.5.2.1 Performance

As mentioned above XFS performs very well when it needs to cater for large files. What all file-systems have in common is that a mix of workload characteristics is often detrimental to the performance of the entire system. XFS provides a `rtdev` and `logdev` configuration option which allows meta-data as well as its logfile to be stored on a different disk or volume than the actual file-data. Devices, either HDD, SSD as well as external arrays can therefore optimise their caching algorithms to cater for these specific workloads.
It is however important to design a proper backend system that allows speedy handling of both workloads in parallel. If, for example, metadata update workloads are not able to keep up there will be a delay on the handling of the file-data as well. Another problem that is often seen is that in SAN environments these workloads are being traversed over the same set of hostbus adapters therefore significantly impacting the traffic-flow and therefore overall performance of the IO stack.

3.5.2.2 mkfs.xfs

Creating the XFS filesystem on any disk or volume is similarly bound by restrictions on the underlying architecture. Optimised alignment to disk-geometry is always recommended. Options like delayed allocation and direct IO should be tested in conjunction with the

applications. An incorrect setting here may not be adjustable after the filesystem is created. Refer to the documentation which options are static or dynamic.

3.5.2.3 Reliability

As with `EXT4`, `XFS` is also a journaling filesystem and has similar recovery algorithms to its arsenal. The naming convention is a bit different where "block groups" are called "allocation groups". Its design principles are more or less the same.

When it comes to adjusting parameters diverting from the defaults you better make sure what you're doing. The XFS FAQ [12] basically indicates not to start turning knobs unless you really know that the application workload is running into issues which are actually caused by the filesystem. The same is true for other filesystems as well. Most of the time the build-in code is able to figure out the best defaults. If you have certain hardware like raid-controllers who mask the layout of the underlying disk configuration you may need to adjust the stripe-width and stripe-size to align with the hardware capabilities but thats about it.

A few options you may take into account. Most of them are optional but may improve performance, reliability, or both. Some settings may, or will, also have some impact on performance so taking this into account before releasing the volume for production is advised.

- Checksumming. CRC32 calculation will be done on all metadata. It is enabled by default and is a prerequisite for some other functions below
  - -m crc=1
- Free inode btree. This is a secondary btree that contains and tracks free inodes. Instead of using the data allocation inodes this allows for faster lookup of free clusters of required sizes. It has no effect on reliability. The crc checksumming will need to be enabled.
  - -m finobt=1
- Reverse mapping btree. Is used as some sort of secondary index of the primary space usage data. It can be used as a cross reference to speed up the repair process and it also provides the option to do this online as corrupted primary metadata can be rebuild from this secondary btree.
  - -m rmapbt=1
- Big Time. You need to have a recent (ie 5.10 or later) kernel for this. It allows for timestamps beyond 2038 to solve the epoch+(2^32) problem.
  - -m bigtime=1
- Access time. Really short: don't touch this and leave this on. The way most filesystems use this is by `relatime` anyway.
- The ftype (or d_type) parameter is a prerequisite for overlay filesystems like the ones used by Docker and other container based abstraction layers. When the crc checksumming is turned on the ftype is set to `1` automatically. Basically what it does is that it stores the `inode type` in the directory structure so that a separate lookup on

the inode of the file is not required.

The disk geometry of the filesystem should be aligned to the underlying raid setup. If your raid controller has created a raid 6 8+2 setup with a stripe size of 256K your XFS filesystem should be made aware of that. The `mkfs.xfs` command should be used with the `-d su=256k sw=8` parameters. Be aware that the `su` value should be a multiple of the `blocksize` value as per the `-b size=xxx` parameter.

```
sserver:~ # mkfs.xfs -f -b size=4k -d agcount=6,su=256k,sw=8 /dev/nvme0n1
meta-data=/dev/nvme0n1           isize=512    agcount=6, agsize=349568 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1
data     =                       bsize=4096   blocks=2097152, imaxpct=25
         =                       sunit=64     swidth=512 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=64 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

To clarify the differences between `su` and `sunit` as well as `sw` and `swidth`, potentially avoid confusion, please take into account the following:

Table 3.4: Stripe Unit and Stripe Width

Unit	Explanation
sunit	This is used to specify the stripe unit for a RAID device or a logical volume. The value has to be specified in 512-byte block units. Use the `su` parameter to specify the stripe unit size in bytes. This parameter ensures that data allocations will be stripe unit aligned when the current end of file is being extended and the file size is larger than 512KiB. Also inode allocations and the internal log will be stripe unit aligned.
su	The su parameter is used to specify the stripe unit for a RAID device or a striped logical volume. The value has to be specified in bytes, (usually using the m or g suffixes). This value must be a multiple of the filesystem block size.

Unit	Explanation
swidth	This is used to specify the stripe width for a RAID device or a striped logical volume. The value has to be specified in 512-byte block units. Use the sw parameter to specify the stripe width size in bytes. This parameter is required if `-d sunit` has been specified and it has to be a multiple of the -d sunit parameter.
sw	Is an alternative to using swidth. The `sw` parameter is used to specify the stripe width for a RAID device or striped logical volume. The value is expressed as a multiplier of the stripe unit, usually the same as the number of stripe members in the logical volume configuration, or data disks in a RAID device.When a filesystem is created on a logical volume device, mkfs.xfs will automatically query the logical volume for appropriate sunit and swidth values.

See the RAID chapter for more information on stripe units and stripe width.

The following shows the correlation between the block size, stripe unit and stripe width. The block size is 4096 bytes which correlates to 8*512 stripe units where therefore a stripewidth of also 1 as I put 8 in the `swidth` parameter. That means that it will exactly fit into 1 stripe unit. Not very helpful.

```
sserver:~ # mkfs.xfs -f -b size=4096 -d agcount=6,sunit=8,swidth=8 /dev/nvme0n1
meta-data=/dev/nvme0n1           isize=512    agcount=6, agsize=349526 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1
data     =                       bsize=4096   blocks=2097152, imaxpct=25
         =                       sunit=1      swidth=1 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

In the next example I use the same parameters except the swidth parameter which is now set at 64. This means it sets 8 blocks of 512 bytes * 8 apart for striping.

```
sserver:~ # mkfs.xfs -f -b size=4096 -d agcount=6,sunit=8,swidth=64 /dev/nvme0n1
meta-data=/dev/nvme0n1           isize=512    agcount=6, agsize=349526 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1
data     =                       bsize=4096   blocks=2097152, imaxpct=25
         =                       sunit=1      swidth=8 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=1 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

If the hardware provides you with a configuration where the chucksize is 256k and a raid6 setup of 8+2 you would set the `su` as 256k and the `sw` to 8 as shown below.

```
sserver:~ # mkfs.xfs -f -b size=4k -d agcount=6,su=256k,sw=8 /dev/nvme0n1
meta-data=/dev/nvme0n1           isize=512    agcount=6, agsize=349568 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1
data     =                       bsize=4096   blocks=2097152, imaxpct=25
         =                       sunit=64     swidth=512 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=64 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

Be aware of the above. This is not only valid for XFS but applies to all filesystem that are created on top of raid configuration. If the raid setup is configured in software via `LVM` or `MD`, most filesystems have fucntionality build in where it can detect the setup and create the alignment dynamically. Some hardware drivers may also provide the same functionality but you would have to consult the manuals of the respective vendor.

3.5.2.4 CoW

XFS supports some sort of Copy on Write mechanism called `reference links`. The mkfs.xfs, by default, enables this upon creation. You can see this with the `xfs_info` (or xfs_db info interactive) command output. The `reflink` tag should show `1`. The reflink option does require crc's to be enabled which subsequently requires a block size of at least 512 bytes.

What is does under the hood is it creates a copy of the btree where the reference counts are kept. As this is very dynamic and can be arranged inside the volume itself without the need for a separate provided volume like `overlayfs` needs.

The section below shows 3 files where one is the original file, one is a copy of a reference

linked file and one is a regular copy.

```
sserver:~/nvme0n1 # ll
total 6291444
-rw-r--r-- 1 root root 2147479552 Jul 19 14:41 2G-orig-file.bin
-rw-r--r-- 1 root root 2147479552 Jul 19 14:47 2G-reflink.bin
-rw-r--r-- 1 root root 2147479552 Jul 19 14:47 2G-regular-cp.bin
```

You can see that all three files occupy exactly the same amount of bytes. Or so it seems as the `2G-reflink.bin` is actually only a copy of the extents pointers in the seperate btree the filesystem created.

We can see this with the `xfs_bmap` tool

```
sserver:~/nvme0n1 # xfs_bmap -v 2G-*

2G-orig-file.bin:
 EXT: FILE-OFFSET      BLOCK-RANGE      AG AG-OFFSET            TOTAL
   0: [0..3999999]:    131264..4131263   0 (131264..4131263) 4000000 100000

2G-reflink.bin:
 EXT: FILE-OFFSET      BLOCK-RANGE      AG AG-OFFSET            TOTAL
   0: [0..3999999]:    131264..4131263   0 (131264..4131263) 4000000 100000

2G-regular-cp.bin:
 EXT: FILE-OFFSET      BLOCK-RANGE      AG AG-OFFSET             TOTAL
   0: [0..3999999]:    4131264..8131263  0 (4131264..8131263) 4000000
```

As shown the occupied space from both the original and the reflinked file point to the same block range whereas the regular copied file is located on different block ranges.

The problem is that we cannot see with a regular `ls -l` output which file is the original and which is the linked one. It also skews the output of `df` and `du`

```
sserver:~/nvme0n1 # df .
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/nvme0n1     8323072 4285384   4037688  52% /root/nvme0n1

sserver:~/nvme0n1 # time cp --reflink 2G-orig-file.bin 2G-reflink.bin

sserver:~/nvme0n1 # df .
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/nvme0n1     8323072 4285384   4037688  52% /root/nvme0n1
```

In both cases the amount of free blocks remains the same in the `df` output whereas in the `du` output the additional 2G is shown.

```
sserver:~/nvme0n1 # du -a
2097148 ./2G-orig-file.bin
2097148 ./2G-regularcp.bin
4194296 .

sserver:~/nvme0n1 # cp --reflink 2G-orig-file.bin 2G-reflink.bin

sserver:~/nvme0n1 # du -a
2097148 ./2G-orig-file.bin
2097148 ./2G-regularcp.bin
2097148 ./2G-reflink.bin
6291444 .
```

This is not a bad thing as it will prevent most tools from reporting values that at some stage may result in an ENOSP [^End of Space] message.

Any action now done on either the reflinked file or the orginal file will result in a CoW action.

```
sserver:~/nvme0n1 # printf "hello" | dd of=./2G-reflink.bin bs=1M count=1 \
    seek=1024 conv=notrunc
0+1 records in
0+1 records out
5 bytes copied, 0.00719305 s, 0.7 kB/s

sserver:~/nvme0n1 # xfs_bmap -v 2G-*
2G-orig-file.bin:
 EXT: FILE-OFFSET      BLOCK-RANGE      AG AG-OFFSET            TOTAL
   0: [0..3999999]:    131264..4131263   0 (131264..4131263) 4000000 100000
2G-reflink.bin:
 EXT: FILE-OFFSET         BLOCK-RANGE      AG AG-OFFSET             TOTAL
   0: [0..2097151]:       131264..2228415   0 (131264..2228415)  2097152 100000
   1: [2097152..2097159]: 8131264..8131271  0 (8131264..8131271)       8
   2: [2097160..3999999]: 2228424..4131263  0 (2228424..4131263) 1902840 100000
2G-regular-cp.bin:
 EXT: FILE-OFFSET      BLOCK-RANGE      AG AG-OFFSET             TOTAL
   0: [0..3999999]:    4131264..8131263  0 (4131264..8131263) 4000000
```

As the change happened in the middle of a 2G file so 2 new extents had to be created.

3.5.2.5 Caution with reflinks

The sheer amount of administrative work that needs to be done when files have a large number of reflinks and therefore extents can cause a huge backlog in IO's when the backend

system is not able to cater for this. A very detailed explanation can be followed in this email thread [13]

An example is when changes happen in database files where continuous updates are made on tables, indices, views, stored procedures etc. The following output shows what happens if just a relatively small change sequence happens on the same two files. 50 writes in the original file at different offsets did not only create new extents in the orginal file but also over 90 new extends for the reflinked file. This is only a small file with a limited range of blocks, but you can imagine that having terabytes of data sitting on very large disks, the amount of administration that is kept in the btrees is very significant.

```
sserver:~/nvme0n1 # for i in {1..50} ; do printf "hello" | dd of=./2G-reflink.bin \
  bs=1M count=1 seek=${RANDOM:0:3} conv=notrunc; done
0+1 records in
0+1 records out

sserver:~/nvme0n1 # xfs_bmap -v 2G-*
2G-orig-file.bin:
 EXT: FILE-OFFSET       BLOCK-RANGE      AG AG-OFFSET         TOTAL
   0: [0..215039]:      131264..346303    0 (131264..346303) 215040 100000
   1: [215040..215047]: 346304..346311    0 (346304..346311)      8
   2: [215048..217087]: 346312..348351    0 (346312..348351)   2040 100000
   3: [217088..217095]: 348352..348359    0 (348352..348359)      8
<snip>
  29: [346112..346119]: 477376..477383    0 (477376..477383)      8
  30: [346120..364543]: 477384..495807    0 (477384..495807)  18424 100000

 2G-reflink.bin:
 EXT: FILE-OFFSET         BLOCK-RANGE      AG AG-OFFSET            TOTAL
   0: [0..215039]:        131264..346303    0 (131264..346303)    215040 100000
   1: [215040..215047]:   8131272..8131279  0 (8131272..8131279)       8
   2: [215048..217087]:   346312..348351    0 (346312..348351)      2040 100000
   3: [217088..217095]:   8131528..8131535  0 (8131528..8131535)       8
<snip>
  92: [1820680..2097151]: 1951944..2228415  0 (1951944..2228415)  276472 100000
  93: [2097152..2097159]: 8131264..8131271  0 (8131264..8131271)       8
  94: [2097160..3999999]: 2228424..4131263  0 (2228424..4131263) 1902840 100000
```

3.5.2.6 Reflinks turned off

If the filesystem is not created with the reference option enabled you will get the error that the operation is not supported.

```
sserver:~/nvme0n1 # dd if=/dev/urandom of=./2G-orig-file.bin bs=2G count=1
0+1 records in
0+1 records out
2147479552 bytes (2.1 GB, 2.0 GiB) copied, 6.45114 s, 333 MB/s

sserver:~/nvme0n1 # cp --reflink 2G-orig-file.bin 2G-reflink.bin
cp: failed to clone '2G-reflink.bin' from '2G-orig-file.bin': Operation not supported

sserver:~/nvme0n1 # xfs_info .
meta-data=/dev/nvme0n1           isize=256    agcount=4, agsize=4194304 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=0        finobt=0, sparse=0, rmapbt=0
         =                       reflink=0    bigtime=0 inobtcount=0
data     =                       bsize=512    blocks=16777216, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=512    blocks=131072, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

3.5.2.7 Information collection

XFS comes with an interactive tool out of the xfstools package called *xfs_db* .

```
xfs_db> info
meta-data=/dev/mapper/mpathd     isize=512    agcount=4, agsize=655360 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=2621440, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
xfs_db>
```

Now, don't knock yourself out on a live filesystem if you don't know 110% what you're doing. The xfs_db tool hacks right into the heart of xfs and can.... ,no no WILL, do much harm without questioning if you type in the wrong commands and press enter. There is no ***"Are you really sure?"*** confirmation question presented to you. A more admin friendly tool is xfs_admin, which under the hood uses xfs_db, and provides a limited set of options to adjust certain parameters in the filesystem. Still don't do anything on a live filesystem with production data resting on it.

On large diskdrives (> 1TB) you may encounter an issue where the filesystem indicates it is full but a `df` shows there is still a fair amount of space available. How is that possible ?!?!?. By default `XFS` internally uses 64bit inode numbers however 32bit hardware platforms and many applications cannot handle the high order 32bits and therefore cannot address inode numbers in that 32 bit high-order space. If the inodes in the lower addressable space are all allocated no new inodes can be created and you'll get the `disk full` scenario. There are basically two ways to prevent that from happening.

- Don't use such large drives (:-))
- Create or mount the filesystem with 64bit inode support.

As many experienced system administrators will tell you it is often better to start with the defaults and adjust where and when required. Trying to tune upfront without knowing what knobs to turn for which occasion will most often have very negative results. That being said if you already know or can safely anticipate the usage and capacity requirements, the file system should already be created with these parameters in mind, An example is the amount of allocation groups you would create.

3.5.2.8 Backup & Recovery

`XFS` provides a recovery utility called `xfs_repair`. It can be used in some sort of query mode with the `-n` parameter basically indicating to not actually change anything on the filesystem. If you suspect corruption this output will tell you in a similar fashion as EXT4 does. As `XFS` repair action will consume a fair amount of memory, depending on the filesystem size, you will need to restrict the use of that especially when the system itself has active applications and tasks doing work on other disks/volumes. The `-m` parameter will allow you to do just that.

`xfs_check` is a second tool that can be used to verify filesystem integrity. This is basically a script that under the hood uses `xfs_db` to scan and verify its structures. It uses a different methodology than `xfs_repair` so both `xfs_repair` and `xfs_check` can be used to cross-check each other. `zfs_repair` runs through 7 stages of different checks each depending on the previous one to complete successfully.

Another utility called `xfsdump` can create a filesystem image or subsets of it and write this to a file or another device such as a tape. It can be used as a "quick&dirty" backup utility. The `xfsrestore` counterpart does obviously the restore of it. Both utilities can be used for incremental backup and restore operations. Be aware that all housekeeping like tape-labeling etc is a manual tasks and your spreadsheet tool will get a good workout if you decide to use this. A nice option on the `xfsrestore` side it the ability to peruse the backup if a `xfsdump` was sent to a file. The `-i` parameter will set the tool into interactive mode where commands like `ls` and `cd` simply work and the `add` and `delete` commands can, indeed, add or remove files and directories from the restore stack. As mentioned this is no alternative for a proper backup mechanism but ideal in test/dev environments where some disk/volume states may need to be stored and recovered relatively frequently.

The recoverability of the filesystem obviously depends on the state of the indexes and journal. If the journal itself is corrupt a replay of that would be of no use and `xfs_repair` or `xfs_check` will show that. The only option is then to clear the journal with `xfs_repair -L` and run a complete check again. There most likely, will be data in the ***lost&found*** folder when inodes are found that do not have a reference in directory tables in the respective allocation groups. The usefulness of that data cannot be determined. Sometimes when a file is smaller than the block-size and is able to be stored with the inode itself you might be able to recover it again. If however the file crosses one or blocks and thus extents the chances are fairly large the content is no longer usable.

3.5.3 BTRFS

All in one filesystem `btrfs` (or `butter fs` or `beetree fs` or `b t r f s`) is a multifunctional filesystem. From a functionality perspective it has much similarities with `ZFS` in the sense it's a volume manager and filesystem in one go. It is very feature rich but the main problem is its "production readiness". The developers have the `btrfs` filesystem in a ***almost ready*** state for over a decade which basically means there are still significant hurdles for it to be considered stable. The project has a dedicated status webpage[14] that outlines the functions and features that are OK or still need work. That is also the likely reason Redhat decided to depreciate the filesystem from its RHEL 6.6, 7.4 and 8 versions. If you still want to use it you will need to collect separate btrfs rpm packages from the web and install them manually. On the flip-side SUSE made `btrfs` its default filesystem at the time of this writing. Other distribution may follow suit.

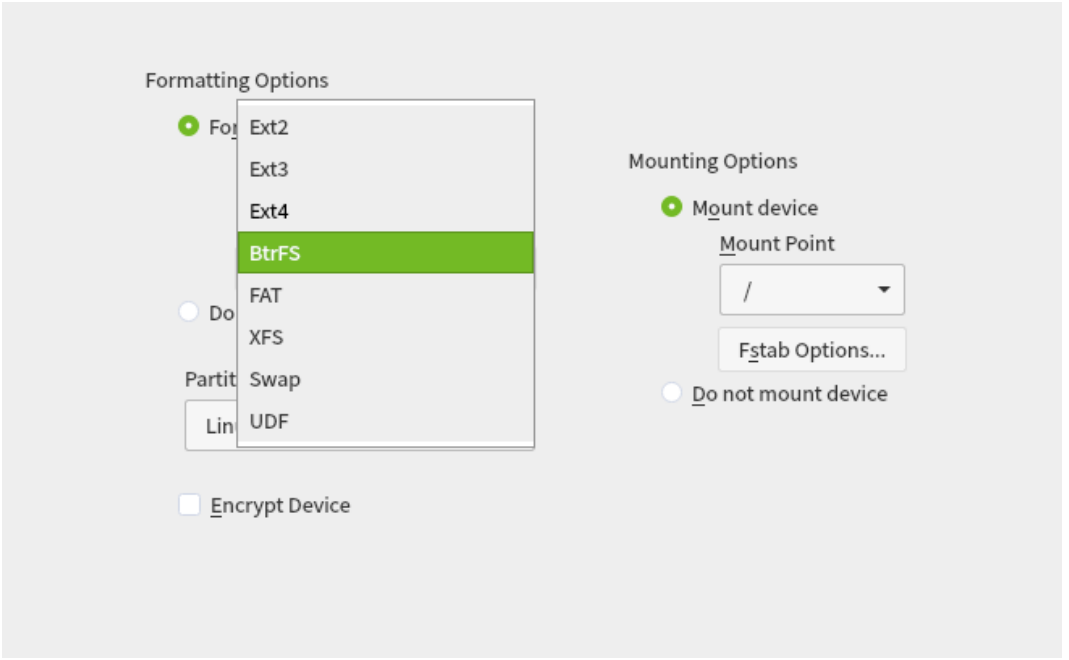


Figure 3.2: SUSE Filesystem Selection

Obviously the choice is yours.

The design thoughts around `btrfs` are fairly much aligned with the ones the Solaris team from SUN had back in the days when they were developing ZFS. Have a single storage management facility taking care of storing and retrieving data with all functionality required to ensure reliability and integrity.

3.5.3.1 Operations

`btrfs` abstracts complexities around physical disks, volume groups and individual volumes and treats underlying disks as a global pool of storage. Based on mkfs.btrfs command line parameters it creates an addressable volume and subvolumes out of these. The redundancy and resiliency comes mainly out of the CoW (Copy on Write) as well as snapshot functionality. Data-blocks can be striped across chunks and are checksummed upon write. This provides self-healing capabilities as when a corrupted blocks fail a checksum its redundant block will be copied to a new/different chunk, its inode updated as well as its index. A physically bad sector can be marked as bad not to be used again. In case of an externally provisioned disk out of an array you will not see this as this is simply resolved on that side.

Depending on your distro you may need to install `btrfs` and the `btrfs` tools. If you manually compile a kernel be sure to flag the `btrfs` kernel either as a module or statically linked. If the boot-device runs `btrfs` be sure to update the initrd with **mkinitrd_** or **dracut**.
Unequal usage of the filesystem amongst drives will happen when there are drives added. There is no dynamic re-balancing when drives are added to the `btrfs` diskpool. This is mainly because it requires a fair amount of work which may impact other production load. The usage dispersion among the drives may be view by executing the `btrfs filesystem usage <mountpoint>` command.

```
opensuse:~ # btrfs filesystem usage /mnt/lst
Overall:
    Device size:    32.00GiB
    Device allocated:     3.56GiB
    Device unallocated:    28.44GiB
    Device missing:       0.00B
    Used:    456.92MiB
    Free (estimated):    10.33GiB (min: 10.33GiB)
    Data ratio:         3.00
    Metadata ratio:        2.00
    Global reserve:     3.25MiB (used: 0.00B)
    Multiple profiles:          no

Data,RAID1C3: Size:1.00GiB, Used:152.11MiB (14.85%)
   /dev/sdb    1.00GiB
   /dev/sdc       1.00GiB
   /dev/sdd       1.00GiB

Metadata,RAID1: Size:256.00MiB, Used:288.00KiB (0.11%)
   /dev/sdb     256.00MiB
   /dev/sde     256.00MiB

System,RAID1: Size:32.00MiB, Used:16.00KiB (0.05%)
   /dev/sdc      32.00MiB
   /dev/sde      32.00MiB

Unallocated:
   /dev/sdb       6.75GiB
   /dev/sdc       6.97GiB
   /dev/sdd       7.00GiB
   /dev/sde       7.72GiB
```

As the layout of the aforementioned filesystem is in a RAID1C3 configuration you would normally expect an even distribution but as you can see the data, meta-data and system

information is not equally balanced in this instance. If you have a mix of very large files and a few small ones may cause this.

Example:

```
opensuse:/mnt/lst # ls -ilh
total 152M
258 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_1.rnd
259 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_2.rnd
260 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_3.rnd
261 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_4.rnd
262 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_5.rnd
263 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_6.rnd
264 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_7.rnd
265 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_8.rnd
266 -rw-r--r-- 1 root root 4.0M May 30 22:59 file_9.rnd
257 -rw-r--r-- 1 root root 117M May 29 23:12 test.rnd
```

The more files you have stored with a different size the more equally the spread of files will be over the filesystem over time. `btrfs` does provide a re-balancing process but it is advised to only execute this during low application utilisation times as it will cause a lot of IO to and from the disks. This can be limited by using filter parameters that allow to select various types like only certain devices, block-ranges, usage-percentages etc. Be aware that the rebalancing operation requires **"work space"** in order to shuffle blocks around. This space is depending on the number of block-groups and extents in those block-groups and not the effective usage rate by the data itself.
The `man 8 btrfs-balance` man page has good and updated information around this topic as well as explanatory examples. It is most often not a real issue as adding devices is fairly straight forward so rebalancing can be executed transparently. Once more it is advised to execute this during maintenance windows or non-business hours. (do these still exist ??)

3.5.3.2 Verifying the filesystem

The `btrfs check` command verifies the integrity of the filesystem. As all data is checksummed the validity of the data itself is continuously checked and therefore the `btrfs check` is mainly checking the structure of the filesystem itself. Be aware that this required the filesystem to be unmounted.

3.5.3.3 Growing the filesystem

When running low on space a `btrfs` filesystem can be dynamically extended. The process itself is not much different when using a single device.

```
localhost:~ # btrfs filesystem df /
Data, single: total=7.48GiB, used=7.48GiB
```

```
System, DUP: total=6.00MiB, used=16.00KiB
Metadata, DUP: total=256.00MiB, used=144.33MiB
GlobalReserve, single: total=15.73MiB, used=0.00B
```

The underlying partition representation looks like this.

```
localhost:~ # parted /dev/sda
GNU Parted 3.4
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Warning: Not all of the space available to /dev/sda appears to be used, you \
  can fix the GPT to use all of the space (an extra 16777216 blocks) or \
  continue with the current setting?
Fix/Ignore? fix
Model: VBOX HARDDISK (scsi)
Disk /dev/sda: 17.2GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags: pmbr_boot

Number  Start   End     Size    File system  Name  Flags
 1      1049kB  9437kB  8389kB                     bios_grub
 2      9437kB  8590MB  8580MB  btrfs              legacy_boot
```

First the partition itself needs to be resized (See the partitions chapter) after which the filesystem can be dynamically be resized. This also works for boot and already mounted filesystems.

```
localhost:~ # btrfs filesystem resize max /
Resize device id 1 (/dev/sda2) from 7.99GiB to max
```

And we're back in business

```
localhost:~ # df /
Filesystem     1K-blocks    Used Available Use% Mounted on
/dev/sda2       15615784 8153772   7236100  53% /
```

3.5.3.4 Failed devices

The dynamics of `btrfs` result in the fact that the inbuilt redundancy takes care of relocating and rebalancing the data over the remaining disks in the filesystem. Prerequisite are obviously the availability of enough space and the redundancy algorithm assigned to the volume. The redundancy algorithm is assigned during the creation of the filesystem with the `-d` parameter. At the time of this writing one of the limitations is the unsupported use

of RAID5 and RAID6 profiles. There are known issues with the write hole which can result in data loss.

There is not really a reason to use R5/6 as the underlying operations with btrfs is done on a block level where data blocks are distributed over the disks in that volume irrespective of the number of disks. So a R1C3 profile will copy a block to 3 independent disks.

Chapter 4

Encryption

One of the most used encryption systems on Linux is done via dm-crypt[15] and LUKS[16] [1]. The `dm-crypt` part is basically the encryption engine, `cryptsetup` [17] the configuration side and `LUKS` the key management portion.

From a storage troubleshooting perspective there isn't much you can do, it either works or it doesn't. If you forget the passwords or lose the LUKS keys your pretty much out a LUKS (:-)). From a performance side it may observe some impact but these are often primarily seen on the CPU side propagating in IO reduction. The dm-crypt abstraction lies entirely in kernel space and is only implementing a very thin layer on the block level. It does not handle individual files or directories. Since it is part of the device-mapper parts in the kernel that layer can be put on any block-device that you can normally directly address. Whether this is sda, hda, USB, software raid, virtual disk doesn't matter.

There are of course numerous other options whether open source, like TrueCrypt, or commercial solutions, like BitLocker I'll leave you with our friends from Google to find these and obtain more information around it.

Example of an encrypted volume:

[1] Linux Unified Key Setup

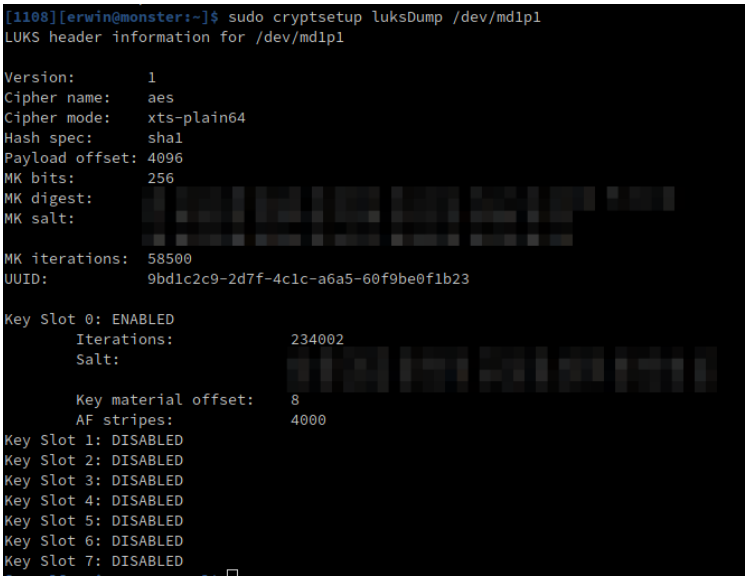


Figure 4.1: Encrypted volume example

4.1 Using dm-crypt, cryptsetup and LUKS

The encryption layer sits right under the file-system layer. It is therefore transparent to any filesystem and has no effect on the underlying addressing schema whether this is a partition, physical disk, multi-path device or logical volume. As seen in the image above the encryption layer sits on the first partition of a raid device. (We'll touch on raid later on).

As with all storage related action be ensured that proper backups are in place and that restores have actually been tested and their result is 100% successful.

Creating a encryption layer on a partition is fairly straightforward:

```
opensuse:~ # cryptsetup luksFormat /dev/sdb1

WARNING!
========
This will overwrite data on /dev/sdb1 irrevocably.

Are you sure? (Type 'yes' in capital letters): YES
Enter passphrase for /dev/sdb1:
Verify passphrase:
```

As the message indicates be aware that everything on that partition is wiped as the entire

space is overwritten. The passphrase is checked against a standard dictionary and some checks are done on entropy but not overly complex. Ensure that you create a password or passphrase that is secure enough to keep a decent size computer system busy for a few years. (:-))

The outcome of the configuration can be viewed like this:

```
opensuse:~ # cryptsetup luksDump /dev/sdb1
LUKS header information for /dev/sdb1

Version:           1
Cipher name:       aes
Cipher mode:       xts-plain64
Hash spec:         sha256
Payload offset:    4096
MK bits:           512
MK digest:         77 bc a2 cd 88 94 38 c7 80 76 30 a3 e4 03 c0 3e 87 a8 39 41
MK salt:           b2 36 78 61 68 aa 00 fc e4 a1 0c 21 4f 2a eb 40
                   37 89 07 91 99 b2 16 ce 39 a1 45 76 0c e1 28 c6
MK iterations:     136391
UUID:              19e50d16-7dd5-46c4-b5e4-977c3d044193

Key Slot 0: ENABLED
    Iterations:             2179990
    Salt:                   e0 bd 7c 4d 8d 34 99 7f 33 3d 43 a3 8c 27 71 44
                              0c 22 7d e3 24 4a da f4 c1 69 7f 31 93 13 48 68
    Key material offset:    8
    AF stripes:                4000
Key Slot 1: DISABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

Depending on the options you use when creating the dm-crypt device one or more values may differ. As soon as the above configuration is created it is not directly usable. What happened is that a "container" is created which needs to be opened first. By using the *"crypsetup luksOpen /dev/xxx <name>"* command, a logical device is created via the device-mapper in a similar fashion as logical volumes by LVM. That device represents the addressable diskspace of that container.

In this case by opening the container you can see the following.

```
opensuse:~ # cryptsetup luksOpen /dev/sdb1 home
Enter passphrase for /dev/sdb1:

opensuse:~ # ls /dev/mapper/
control  home
```

Now just create a filesystem, mount the device as per normal (see filesystem chapter).

```
opensuse:~ # mkfs.ext4 /dev/mapper/home
mke2fs 1.45.6 (20-Mar-2020)
Creating filesystem with 523776 4k blocks and 131072 inodes
Filesystem UUID: c9ede0c1-3d53-4729-af78-d17126ebc32f
Superblock backups stored on blocks:
.....32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (8192 blocks): done
Writing superblocks and filesystem accounting information: done

opensuse:~ # mount /dev/mapper/home /encvol/
opensuse:~ # ls /encvol/
lost+found
opensuse:~ #
```

The challenge comes when a container closes. It basically is analogues to locking the door and keep the key in a safe place.

First we unmount the volume after which we manually close the container. You'll find the device-mapper entry is now removed and the only thing that remains is the raw `/dev/sdb` device with the partition layout.

```
opensuse:/encvol # cd /
opensuse:/ # umount /encvol

opensuse:/ # ls /dev/mapper/
control  home

opensuse:/ # cryptsetup luksClose home

opensuse:/ # ls /dev/mapper/
control

opensuse:/ #
```

The underlying disk and partition tables do not have any reference to the dm-crypt container or filesystem that resides on them.

```
opensuse:/ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: HARDDISK
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 85A99410-1BF8-4E2C-A1F9-B2B5A6121373
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 8388541 sectors (4.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         4196351   2.0 GiB     8300  Linux filesystem
   2         4196352         8390655   2.0 GiB     8302  Linux /home

Command (? for help):
```

A way to make this a bit more intelligent and easier for administration purposes we can change the partition type to "8308" which then refers to a dm-crypt volume. See lines 5 and 19 below.

```
1  Command (? for help): t
2  Partition number (1-2): 1
3  Current type is 8300 (Linux filesystem)
4  Hex code or GUID (L to show codes, Enter = 8300): 8308
5  Changed type of partition to 'Linux dm-crypt'
6
7  Command (? for help): p
8  Disk /dev/sdb: 16777216 sectors, 8.0 GiB
9  Model: HARDDISK
```

```
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 85A99410-1BF8-4E2C-A1F9-B2B5A6121373
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 8388541 sectors (4.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         4196351   2.0 GiB     8308  Linux dm-crypt
   2         4196352         8390655   2.0 GiB     8302  Linux /home

Command (? for help):
```

4.2 Backup and recovery

The main difference between filesystem, partion managers, logical volumes and encryption is that the encryption layer directly changes the data how it is stored on disk. That is the intention of encryption. No higher or lower level interfaces, tools or methodologies shall be able to make any use of the content without having the proper authorisation and authentication therefore enabling to read and modify that data in a meaningful way.
When volume or partition managers and even filesystems get into a corrupted state the tools provided may be able (in a fair few occasions that is) to recover and move on providing the higher layer applications the data and storage space they need. Things are much different with encryption. If keys are lost or encryption headers are corrupted all data in that container is permanently lost. The only way to recover the data is from a recent and valid backup.

Fortunately dm-crypt provide a way to backup the LUKS header to a file.

```
opensuse:~ # cryptsetup luksHeaderBackup --header-backup-file=luksbackup.hdr /dev/sdb1
opensuse:~ # ls -l luksbackup.hdr
-r-------- 1 root root 2068480 Aug  7 16:24 luksbackup.hdr
```

As shown the file has very limited permissions by default. Ensure this file is copied and stored at least twice in separate vaults somewhere far away so that other problems like media defects, fires, flooding or even malicious actions have a lesser effect in case you'd need it again.

The first 256 bytes of the header provide some readable data when shown in a hex reader:

```
00000000  4c 55 4b 53 ba be 00 01  61 65 73 00 00 00 00 00  |LUKS....aes.....|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000020  00 00 00 00 00 00 00 00  78 74 73 2d 70 6c 61 69  |........xts-plai|
00000030  6e 36 34 00 00 00 00 00  00 00 00 00 00 00 00 00  |n64.............|
```

```
00000040  00 00 00 00 00 00 00 00  73 68 61 32 35 36 00 00  |........sha256..|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000060  00 00 00 00 00 00 00 00  00 00 10 00 00 00 00 40  |...............@|
00000070  17 1c 82 19 bd 0c 6c 0d  22 05 1f 56 72 d0 34 3e  |......l."..Vr.4>|
00000080  1b be 66 2e 62 7f 67 d8  b8 1f ef 19 79 5f 0a 76  |..f.b.g.....y_.v|
00000090  c4 2c c9 9f 03 c1 b2 53  a3 ca 3a b9 88 61 31 82  |.,.....S..:..a1.|
000000a0  4a 8f e9 a6 00 01 ff 7d  65 65 65 35 64 65 30 64  |J......}eee5de0d|
000000b0  2d 31 65 38 39 2d 34 30  38 35 2d 39 61 62 34 2d  |-1e89-4085-9ab4-|
000000c0  37 39 31 62 61 33 31 39  63 33 32 62 00 00 00 00  |791ba319c32b....|
000000d0  00 ac 71 f3 00 21 5e 3a  63 d0 5a 57 38 bc 16 e5  |..q..!^:c.ZW8...|
000000e0  bc df 2b b5 4c b6 2a 94  aa d2 eb 8b 31 4f ba 19  |..+.L.*.....1O..|
000000f0  99 68 97 ae 15 12 ef 03  00 00 00 08 00 00 0f a0  |.h..............|
```

This is also the part that is used by dm-crypt to be able to identify what parameters is should use in order to be able to decrypt the data. The remainder of the header contains the rest of the parameters such as keys etc. (For more information check the dm-crypt[15] website).

Whatever you do, ***do NOT save this file inside the encrypted container***. If you ever shut the door of your car with the keys still inside you more or less can guess what the result is. The only difference is that there is NO WAY to get that file back in case the header gets corrupted.
Compare it to the car with the keys inside but instead of calling road-side assistance you need to call a tow-truck and have it delivered to a wrecking yard. You will have created your own ransomware attack without any possibility to get things back without a recent backup.

Below the disaster:

```
opensuse:~ # dd if=/dev/zero of=/dev/sdb1 count=256 bs=512
256+0 records in
256+0 records out
131072 bytes (131 kB, 128 KiB) copied, 0.00955386 s, 13.7 MB/s

opensuse:~ # cryptsetup luksOpen /dev/sdb1 home
Device /dev/sdb1 is not a valid LUKS device.

opensuse:~ # mount /dev/sdb1 /encvol/
mount: /encvol: wrong fs type, bad option, bad superblock on /dev/sdb1, missing codepage \
   or helper program, or other error.

opensuse:~ #
```

As of this stage the data that resided on that partition is lost if no valid backup was available.

Restoring the header will save your day:

```
opensuse:~ # cryptsetup luksHeaderRestore --header-backup-file=luksbackup.hdr /dev/sdb1

WARNING!
========
Device /dev/sdb1 does not contain LUKS header. Replacing header can destroy \
   data on that device.

Are you sure? (Type 'yes' in capital letters): YES

opensuse:~ # cryptsetup luksOpen /dev/sdb1 home
Enter passphrase for /dev/sdb1:

opensuse:~ # mount /dev/mapper/home /encvol/

opensuse:~ # ls /encvol/
lost+found  thisisatstfile.tst

opensuse:~ # cat /encvol/thisisatstfile.tst
Hello,

This file resides on an encrypted filesystem

opensuse:~ #
```

As mentioned before, if the problem is more widespread and large parts of the disk itself are corrupted you will inevitably lose data. The only way to recover from that is to restore from a full backup.

Chapter 5

Partitioning

As highlighted in the introductions chapter using partitions and volume managers can provide a vast amount of flexibility in storing data and moving it around. By using this abstraction layer a very strict regime of administration between the storage and OS teams need to be adhered to. Mistakes in disk-provisioning out of arrays and having these incorrectly mapped can have significant implications on performance.

The mistake that is most often made is that partitions or luns presented out of a single disk or array pool respectively are aggregated again in virtual volumes.

From a functionality perspective this may make very good sense as it can provide a lot of flexibility in moving data around for what purpose you can think of. It can however at some stage cause a divide & conquer behaviour where parallel IO requests towards the volume get split across the partitions subsequently ending up on the same underlying disk or pool. Depending on the capabilities of the that disk you may observe an increase in queuing behaviour resulting in bottlenecks and delays. It is therefore imperative that design, configuration and ongoing operations are planned meticulously to avoid these kind of things to happen.

5.1 Partitions

Partitions are a method of dividing a single disk into multiple logical disks. This only has an administrative benefit to be able to segregate data on different logical places on the same disk. On workstation and server environments you'll often seen that the OS, swap and temp-spaces are stored on different partitions. The underlying thought being that one process would not be able to go "rogue" and completely fill a partition causing panics due to disk space issues.

Partitions come in various flavours in the sense that the layout of the partition table can be used for specific purposes. When issuing the fdisk command and enter the `l` (Lower case L) you'll see the list of partitions that the fdisk tool supports:

```
Command (m for help): l

 0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
 1  FAT12           27  Hidden NTFS Win 82  Linux swap / So c1  DRDOS/sec (FAT-
 2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden or  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
 5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS data
 6  FAT16           42  SFS             87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell Utility
 8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi ea  Linux extended
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD         eb  BeOS fs
 f  W95 Ext-d (LBA) 54  OnTrackDM6      a6  OpenBSD         ee  GPT
10  OPUS            55  EZ-Drive        a7  NeXTSTEP        ef  EFI (FAT-12/16/
11  Hidden FAT12    56  Golden Bow      a8  Darwin UFS      f0  Linux/PA-RISC b
12  Compaq diagnost 5c  Priam Edisk     a9  NetBSD          f1  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       ab  Darwin boot     f4  SpeedStor
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+      f2  DOS secondary
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fb  VMware VMFS
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fc  VMware VMKCORE
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fd  Linux raid auto
1c  Hidden W95 FAT3 75  PC/IX           bc  Acronis FAT32 L fe  LANstep
1e  Hidden W95 FAT1 80  Old Minix       be  Solaris boot    ff  BBT
```

This may seem overwhelming but in 99.999% of all cases you'de would only need the `GPT` table which stands for `GUID Partition Table`. The layout of the partition table needs to be aligned with the sector size of the physical disk. What this means is that a disk is itself formatted in so called sectors. The old rotational disks had a sector size of 512 bytes and newer disks have 4096 byte sector sizes. A partition that is not aligned properly to these, on disk, sector sizes require additional handling of either the spindle actuator or processing time in the disks' processor/fpga to request the data from the correct blocks.

As an example if a spindle (ie non-flash drive) is showing a block layout of 4096 bytes starting at offset 1 where the partitioning software if using 4096 bytes starting at offset 0, a read IO to the sector starting at 0 will also require the entire second sector to be read as the last byte of sector 0 is not read from the physical disk. That means that not only two reads from

disk are needed but almost the entire block from sector 1 can be disregarded.

If you're running a recent version of Linux than don't worry, this has all been fleshed out and the tools themselves have a fair amount of intelligence in them to handle this. The `parted` tool for example allows for the following:

```
-a alignment-type, --align alignment-type
       Set alignment for newly created partitions, valid alignment types are:

       none   Use the minimum alignment allowed by the disk type.

       cylinder
              Align partitions to cylinders.

       minimal
              Use minimum alignment as given by the disk topology information.
              This and the opt value will use layout information provided by the disk
              to align the logical partition table addresses to actual physical blocks
              on the  disks. The  "min" value  is  the minimum alignment needed to align
              the partition properly to physical blocks, which avoids performance
              degradation.

       optimal
              Use optimum alignment as given by the disk topology information.
              This aligns to a multiple of the  physical block size in a way
              that guarantees optimal performance.
```

Many references in storage documentation show `CHS` (Cylinder - Head - Sector) addresses. In the "good-ol'e-days" the disk actuator was directly addressed by the operating system to steer the head in a certain cylinder above a specific sector and either read from or write to that sector. This has been abandoned in the open-systems world for a long time as the disk firmware takes care of all that and the addressing is now done on a `LBA` (Logical Block Address) level . Modern (Being a flexible term, as in the ones from 2000 and newer :-) .) disks primarily provide a range of space with an array of other capabilities they might have. The handling of the hardware is all done in firmware on that disk.

So unless you manually start adjusting the layout of the partition and configure different off-sets and sector sizes the tools these days do not have much of an issue getting the right information from the disk and create the one that is needed.

5.1.1 MBR

I heard about MBR. What is that ?

The `MBR` or "Master Boot Record" used the first sector of a disk and outlines the partition

table of how that disk is carved up. The MBR was developed and heavily used in the 80-ties mainly by Microsoft DOS and Windows systems up to NT.
The limitation in this was that disks in those days used a 512 byte sector size and the MBR record was only 32 bits long. That meant the maximum drive capacity was 2 TB (2^32 * 512 bytes). When disk drive capacities were in the MB and GB ranges that did not really matter but during the early 90-ies the drives were scratching that 2TB mark and so Intel developed a new concept called UEFI. (Unified Extensible Firmware Interface) of which the GPT was one part.
UEFI was a replacement of the BIOS (Basic Input/Output System) that allowed firmware extensions to be written by 3rd parties to interface the hardware and the Operating System.

For backward compatibility with MBR the GPT layout does not start at sector 0 but at sector 1. This prevented tools that were not GPT compatible/aware from overwriting a GPT partition table. The MBR record in sector 0 only contains a table entry with the 0xEE partition type which basically indicates to operating systems that are not GPT aware that the entire disk is filled up with one single partition of an unknown type. This should, with the emphasis on "should", give partitioning tools that were/are not GPT aware enough information to at least indicate that a partition has used the entire drive and therefore will not overwrite it entirely or partially.

The below snippet shows the MBR information when a disk has a protective MBR indicating the remainder is a partition occupying the entire disk.

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): r

Recovery/transformation command (? for help): o

Disk size is 16777216 sectors (8.0 GiB)
MBR disk identifier: 0x00000000
MBR partitions:

Number  Boot  Start Sector   End Sector   Status      Code
   1                    1     16777215   primary     0xEE   <<<<<<<<<<<<
```

```
Recovery/transformation command (? for help):
```

The second version of MBR is a hybrid form of MBR and GPT where the systems' BIOS still boots from the MBR but the operating system takes over from GPT. This is often seen on systems where UEFI is not used. Being realistic this is in current days only seen on PC level systems and not on any PRO level workstation or server.

5.1.2 GPT

As mentioned above the GUID Partition Table is the most used partitioning system on x86/x86_64 platforms. The GUID part of "GPT" is a vast improvement over MBR as this can be used by a vast array of utilities to uniquely identify a partition.

The GPT partitions have different type GUID's. These are fixed GUID's providing operating systems the ability to identify the exact type of partition this is so that the OS itself does not have to spawn discovery processes. It also provides the ability for multi-OS systems to identify the partition belonging to a different OS type. For example a Linux OS may see a GPT partition with GUID type [18] 6A87C46F-1DD2-11B2-99A6-080020736631 and it immediately knows that it is a Solaris/Illumos swap partition.

The above shows a very flexible and extensible way of addressing and identifying the partitions but it does not inherently have safeguards in place that prevent redundancy in case a partition table goes corrupt or is accidentally overwritten by low-level tools (like for example "dd"). It is therefore imperative that partition tables get backed-up and safeguarded.

The type of system in use for booting (BIOS or EFI) is very important. EFI requires a so called ESP which stands for EFI System partition. This is a small FAT32 partition which houses the information UEFI needs to further boot the system. That boot-loader may well be on a USB drive or another block storage facility.

5.1.3 Creating partitions

There are mainly two tools (outside of the GUI ones) that handle the creation of the partitions. Parted and fdisk are the ones that are mostly used. I leave it up to the documentation of each tool to provide more information. The gdisk, sgdisk and cgdisk tools which I use below were written by Rod Smith [19]. These are more flexible and have some very useful options that the standard tools do not have.

5.1.4 Investigating GPT partitions

The way to view, manipulate, backup and recover a GPT partition is called gdisk which, according to Rod is the GPT fdisk tool. As always be extremely careful with such low-level tools as they can do as much harm as they can do good in case they are incorrectly used.

A GPT record only shows the standard disk information and the GPT identifiers without any partition information if no partition was created. An example:

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: HARDDISK
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 8315DB0B-3BA1-7446-ABB9-CE7E632AA6AE
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 2048, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 16775135 sectors (8.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name

Command (? for help):
```

When a backup is created with the b option of gdisk (see below) the hex-output looks a bit like this:

```
opensuse:~ # hexdump -C gpt-backup.gpt
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001c0  02 00 ee ff ff ff 01 00  00 00 ff ff ff 00 00 00  |................|
000001d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 aa  |..............U.|
00000200  45 46 49 20 50 41 52 54  00 00 01 00 5c 00 00 00  |EFI PART....\...|
00000210  39 ef d9 25 00 00 00 00  01 00 00 00 00 00 00 00  |9..%............|
00000220  ff ff ff 00 00 00 00 00  00 08 00 00 00 00 00 00  |................|
00000230  de ff ff 00 00 00 00 00  0b db 15 83 a1 3b 46 74  |.............;Ft|
00000240  ab b9 ce 7e 63 2a a6 ae  02 00 00 00 00 00 00 00  |...~c*..........|
```

```
00000250  80 00 00 00 80 00 00 00  86 d2 54 ab 00 00 00 00  |..........T.....|
00000260  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  45 46 49 20 50 41 52 54  00 00 01 00 5c 00 00 00  |EFI PART....\...|
00000410  f5 02 df 16 00 00 00 00  ff ff ff 00 00 00 00 00  |................|
00000420  01 00 00 00 00 00 00 00  00 08 00 00 00 00 00 00  |................|
00000430  de ff ff 00 00 00 00 00  0b db 15 83 a1 3b 46 74  |.............;Ft|
00000440  ab b9 ce 7e 63 2a a6 ae  df ff ff 00 00 00 00 00  |...~c*..........|
00000450  80 00 00 00 80 00 00 00  86 d2 54 ab 00 00 00 00  |..........T.....|
00000460  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004600
opensuse:~ #
```

The same backup file can be generated with the `sgdisk` utility in a single command:

```
opensuse:~ # sgdisk -b gpt-backup.bck /dev/sdb
The operation has completed successfully.
```

After creating a new 2 gig partition with a 8300 (Linux FS partition) type the output shows:

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: HARDDISK
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 8315DB0B-3BA1-7446-ABB9-CE7E632AA6AE
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 2048, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 12580831 sectors (6.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
```

```
   1            2048         4196351   2.0 GiB     8300  Linux filesystem

Command (? for help): b
Enter backup filename to save: gpt-backup-gpt
The operation has completed successfully.
```

The backup file shows that :

```
opensuse:~ # hexdump -C gpt-backup-gpt
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001c0  02 00 ee ff ff ff 01 00  00 00 ff ff ff 00 00 00  |................|
000001d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 aa  |..............U.|
00000200  45 46 49 20 50 41 52 54  00 00 01 00 5c 00 00 00  |EFI PART....\...|
00000210  4e 98 c4 1a 00 00 00 00  01 00 00 00 00 00 00 00  |N...............|
00000220  ff ff ff 00 00 00 00 00  00 08 00 00 00 00 00 00  |................|
00000230  de ff ff 00 00 00 00 00  0b db 15 83 a1 3b 46 74  |.............;Ft|
00000240  ab b9 ce 7e 63 2a a6 ae  02 00 00 00 00 00 00 00  |...~c*..........|
00000250  80 00 00 00 80 00 00 00  10 7c 8a fb 00 00 00 00  |.........|......|
00000260  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000400  45 46 49 20 50 41 52 54  00 00 01 00 5c 00 00 00  |EFI PART....\...|
00000410  82 75 c2 29 00 00 00 00  ff ff ff 00 00 00 00 00  |.u.)............|
00000420  01 00 00 00 00 00 00 00  00 08 00 00 00 00 00 00  |................|
00000430  de ff ff 00 00 00 00 00  0b db 15 83 a1 3b 46 74  |.............;Ft|
00000440  ab b9 ce 7e 63 2a a6 ae  df ff ff 00 00 00 00 00  |...~c*..........|
00000450  80 00 00 00 80 00 00 00  10 7c 8a fb 00 00 00 00  |.........|......|
00000460  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00000600  af 3d c6 0f 83 84 72 47  8e 79 3d 69 d8 47 7d e4  |.=....rG.y=i.G}.|
00000610  2c 9c 58 56 91 74 58 4b  b0 3a 03 59 df 31 7e 2b  |,.XV.tXK.:.Y.1~+|
00000620  00 08 00 00 00 00 00 00  ff 07 40 00 00 00 00 00  |..........@.....|
00000630  00 00 00 00 00 00 00 00  4c 00 69 00 6e 00 75 00  |........L.i.n.u.|
00000640  78 00 20 00 66 00 69 00  6c 00 65 00 73 00 79 00  |x. .f.i.l.e.s.y.|
00000650  73 00 74 00 65 00 6d 00  00 00 00 00 00 00 00 00  |s.t.e.m.........|
00000660  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
00004600
```

As you can see from byte offset 0x600 to 0x660 the additional partition is added to the table and the pointers where the GPT partition type is 8300 (ie standard Linux filesystem partition)

The GUID type is shown in the fist line of the partition header. It may seem somewhat confusing but that is because the GPT UID type is stored in mixed endian [20] format (bytes 3-0 , 5-4 , 7-6 , 8-15)

```
00000600  af 3d c6 0f 83 84 72 47  8e 79 3d 69 d8 47 7d e4  |.=....rG.y=i.G}.|
```

GUID code: 0F C6 3D AF 84 83 47 72 8E 79 3D 69 D8 47 7D E4 (Linux filesystem)

As mentioned before the GPT partitioning schema is part of the UEFI specification. [21]

5.2 Partition corruption

In case a corruption of the partition table is observed there is an option to recover from a secondary location on the disk if that location is not corrupted as well. The below shows a corrupted primary partition table whereby a recover option is provided.

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Caution: invalid main GPT header, but valid backup; regenerating main header
from backup!

Warning: Invalid CRC on main header data; loaded backup partition table.
Warning! One or more CRCs don't match. You should repair the disk!
Main header: ERROR
Backup header: OK
Main partition table: OK
Backup partition table: OK

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: damaged

Found invalid MBR and corrupt GPT. What do you want to do? (Using the
GPT MAY permit recovery of GPT data.)
 1 - Use current GPT
 2 - Create blank GPT

Your answer: 1

Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
```

```
Model: HARDDISK
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 8315DB0B-3BA1-7446-ABB9-CE7E632AA6AE
Partition table holds up to 128 entries
Main partition table begins at sector 2016 and ends at sector 2047
First usable sector is 2048, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 12171231 sectors (5.8 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         4196351   2.0 GiB     8300  Linux filesystem
   2         4196352         4605951   200.0 MiB   8302  Linux /home

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/sdb.
The operation has completed successfully.
```

5.3 Recovering partition information

The best way to recover is if a recent backup is available, and it is certain the partition table has not been modified since that backup, a blank GPT can be created followed by a recover->load from file sequence. As shown above the `gdisk` tool verifies the integrity of the backup table via a crc.

The below example shows a similar issue where just the MBR is corrupt. The gdisk tool can dynamically write a new MBR.

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with corrupt MBR; using GPT and will write new
protective MBR on save.
```

```
Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/sdb.
The operation has completed successfully.
```

The result is that the MBR is back in place again.

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help):
```

5.3.1 Full example GPT restoration

The below shows and entire sequence of creating two partitions on an empty disk, making a backup of the partition information subsequently followed by deliberately corrupting the partition table (both primary and backup). This then leaves an un-addressable partition layout. After that we restore the partition layout from file as previously mentioned.

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries in memory.

Command (? for help): n
```

```
Partition number (1-128, default 1):
First sector (34-16777182, default = 2048) or {+-}size{KMGTP}:
Last sector (2048-16777182, default = 16777182) or {+-}size{KMGTP}: +2G
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300):
Changed type of partition to 'Linux filesystem'

Command (? for help): n
Partition number (2-128, default 2):
First sector (34-16777182, default = 4196352) or {+-}size{KMGTP}:
Last sector (4196352-16777182, default = 16777182) or {+-}size{KMGTP}: +2G
Current type is 8300 (Linux filesystem)
Hex code or GUID (L to show codes, Enter = 8300): 8302
Changed type of partition to 'Linux /home'

Command (? for help): w

Final checks complete. About to write GPT data. THIS WILL OVERWRITE EXISTING
PARTITIONS!!

Do you want to proceed? (Y/N): y
OK; writing new GUID partition table (GPT) to /dev/sdb.
The operation has completed successfully.
```

Validation of the partitions correctly written to disk.

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: HARDDISK
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): EC779E82-6FA3-445B-9634-C266739F38C4
Partition table holds up to 128 entries
```

```
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 8388541 sectors (4.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         4196351   2.0 GiB     8300  Linux filesystem
   2         4196352         8390655   2.0 GiB     8302  Linux /home

Command (? for help): q
```

Now we're creating the backup. In the same way an encryption header should be saved elsewhere, also ensure that the backup file of a partition is safeguarded on multiple locations and never on the partition that is reflected in the backup. This may seem obvious but I would've been fairly rich if I received a dollar for each of these cases I encountered.

```
opensuse:~ # sgdisk -b gpt-backup.bck /dev/sdb
The operation has completed successfully.
```

Wiping the partition information by using `dd` and zeroing the first and last sectors (MBR/GPT main and backup tables).

```
opensuse:~ # dd if=/dev/zero of=/dev/sdb count=128 bs=512 seek=0
128+0 records in
128+0 records out
65536 bytes (66 kB, 64 KiB) copied, 0.00702359 s, 9.3 MB/s
opensuse:~ # dd if=/dev/zero of=/dev/sdb count=128 bs=512 seek=16777088
128+0 records in
128+0 records out
65536 bytes (66 kB, 64 KiB) copied, 0.00416024 s, 15.8 MB/s
```

As shown below the information is deleted.

```
opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: not present
  BSD: not present
  APM: not present
  GPT: not present

Creating new GPT entries in memory.

Command (? for help): p
```

```
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: HARDDISK
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 5BF00CBC-C059-4728-8D43-442468CF67A8
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 16777149 sectors (8.0 GiB)

Number  Start (sector)    End (sector)  Size       Code  Name

Command (? for help):
```

Restoring the partition table from backup

```
opensuse:~ # sgdisk -l gpt-backup.bck /dev/sdb
Creating new GPT entries in memory.
The operation has completed successfully.


opensuse:~ # gdisk /dev/sdb
GPT fdisk (gdisk) version 1.0.5

Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present

Found valid GPT with protective MBR; using GPT.

Command (? for help): p
Disk /dev/sdb: 16777216 sectors, 8.0 GiB
Model: HARDDISK
Sector size (logical/physical): 512/512 bytes
Disk identifier (GUID): 85A99410-1BF8-4E2C-A1F9-B2B5A6121373
Partition table holds up to 128 entries
Main partition table begins at sector 2 and ends at sector 33
First usable sector is 34, last usable sector is 16777182
Partitions will be aligned on 2048-sector boundaries
Total free space is 8388541 sectors (4.0 GiB)
```

```
Number  Start (sector)    End (sector)  Size       Code  Name
   1            2048         4196351   2.0 GiB     8300  Linux filesystem
   2         4196352         8390655   2.0 GiB     8302  Linux /home

Command (? for help):
```

And we're back in business.

Be aware that the validity of the partition information itself is only as good as the time and accuracy of the backup. If adjustments are made after a backup is made obviously a restore may fail or will not result in the outcome that was expected.

Another common problem is that disks presented out of storage arrays can be dynamically re-sized. This therefore results in the fact that the backup partition information is no longer located at the end of the address space as required by GPT. This can also be adjusted by rebuilding the GPT tables by using the `gdisk` recovery/transformation "e" option. This will recreate the backup from main and put it at the end of the disk again.

Chapter 6

Volume Managers

Volume managers are most often used to be able to create flexibility on a host system to create logical volumes and be able to resize, move, remap and other useful things. The LVM stack has been a part of the Linux storage subsystem for a long time and has been well maintained and it's function and features are vast. In the next chapter we're going to speak about RAID in the form of the dedicated raid device driver however LVM itself as a volume manager also has some RAID functionality build in. LVM uses some functionality from the `md` driver when it comes to that. I'll highlight this a bit more in that chapter.

As I mentioned Linux LVM(2) is a beast of a tool. I'm not going into the operational side as the online help of many userspace tools is very clear. The below is just for showing the basics and show issues that are fairly common.

6.1 LVM

LVM is part of the Device Mapper stack. The device mapper is a modular group of "tools" which can be used to do many different functions in the IO stack. Managing logical volumes is just one part of it. Features like caching, thin-provisioning, multipathing, raid etc are all part of the device mapper layer. I'll touch on them in subsequent chapters.

```
opensuse:~ # lsblk
NAME     MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda        8:0    0    8G  0 disk
|-sda1     8:1    0    8M  0 part
|-sda2     8:2    0  6.5G  0 part /
|-sda3     8:3    0  1.5G  0 part [SWAP]
sdb        8:16   0    8G  0 disk
```

```
sdc         8:32   0    8G  0 disk
sdd         8:48   0    8G  0 disk
sde         8:64   0    8G  0 disk
sr0        11:0    1 57.8M  0 rom
pktcdvd0 253:0    1 57.8M  0 disk
```

The above shows 4 disks (sdb, sdc, sdd and sde) which I'll use for creating logical volumes. Again, be aware that using any form of abstraction layer does not alter the behaviour of the hardware. Incorrect configuration or design will, not may, seriously hamper functionality and performance. It is of the utmost importance that a layout defined in software matches the capabilities of the hardware.

```
opensuse:~ # pvscan
  No matching physical volumes found

opensuse:~ # pvcreate /dev/sdb
  Physical volume "/dev/sdb" successfully created.
opensuse:~ # pvcreate /dev/sdc
  Physical volume "/dev/sdc" successfully created.
opensuse:~ # pvcreate /dev/sdd
  Physical volume "/dev/sdd" successfully created.
opensuse:~ # pvcreate /dev/sde
  Physical volume "/dev/sde" successfully created.

opensuse:~ # pvscan
  PV /dev/sdb                      lvm2 [8.00 GiB]
  PV /dev/sdc                      lvm2 [8.00 GiB]
  PV /dev/sdd                      lvm2 [8.00 GiB]
  PV /dev/sde                      lvm2 [8.00 GiB]
  Total: 4 [32.00 GiB] / in use: 0 [0   ] / in no VG: 4 [32.00 GiB]
opensuse:~ #
```

LVM uses a layered representation of devices. At the lowest level it's the physical device which it will label as a physical volume. The disk itself is not altered except a small `lvm2` disk-label is written to the device. Of these PV's, so called, Volume Groups (VG) are created.

```
opensuse:~ # vgcreate LST0 /dev/sdb /dev/sdc
  Volume group "LST0" successfully created
opensuse:~ # vgcreate LST1 /dev/sdd /dev/sde
  Volume group "LST1" successfully created

opensuse:~ # pvscan
  PV /dev/sdd   VG LST1            lvm2 [8.00 GiB / 8.00 GiB free]
```

```
  PV /dev/sde   VG LST1            lvm2 [8.00 GiB / 8.00 GiB free]
  PV /dev/sdb   VG LST0            lvm2 [8.00 GiB / 8.00 GiB free]
  PV /dev/sdc   VG LST0            lvm2 [8.00 GiB / 8.00 GiB free]
  Total: 4 [31.98 GiB] / in use: 4 [31.98 GiB] / in no VG: 0 [0   ]
```

As shown we now have two VG's (LST0 and LST1) of which we can carve volumes.

```
opensuse:~ # lvcreate LST0 -L 5G
  Logical volume "lvol0" created.
opensuse:~ # lvcreate LST0 -L 1G
  Logical volume "lvol1" created.

opensuse:~ # ls /dev/LST0/
lvol0  lvol1
```

The volumes shown on `/dev/LST0/` can now be formatted with the filesystem required and used as such.

```
opensuse:~ # mkfs.xfs /dev/LST0/lvol0
meta-data=/dev/LST0/lvol0        isize=512    agcount=4, agsize=327680 blks
         =                       sectsz=512   attr=2, projid32bit=1
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1
data     =                       bsize=4096   blocks=1310720, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0


opensuse:~ # mount /dev/LST0/lvol0 /mnt/lvol0/

opensuse:~ # mount
<snip>
/dev/mapper/LST0-lvol0 on /mnt/lvol0 type xfs (rw,relatime,attr2,inode64,logbufs=8 \
  ,logbsize=32k,noquota)
```

The above volume (lvol0) has a 5GB size and the LVM handler can very easily assign additional space.

```
opensuse:~ # lvresize -L +1G /dev/LST0/lvol0
  Size of logical volume LST0/lvol0 changed from 5.00 GiB (1280 extents) \
    to 6.00 GiB (1536 extents).
  Logical volume LST0/lvol0 successfully resized.
```

6.2 What can go wrong

Lots. :-)

Kidding aside. As you can see the flexibility of LVM in addition to the simplicity makes it very easy to make mistakes. The majority of issues are related to two things, missing disks or performance problems. From a configuration perspective LVM is fairly picky in what it accepts as physical volumes. That being said, it will check if a volume does not have pre-existing partitions and filesystems already sitting on it. The lazy (or over-confident) admin is very quick in finding the `-ff` parameter which will force the action irrespective of what `pvcreate` or `vgcreate` finds.

6.3 PV - Physical Volumes

The physical volume is basically the building block of the LVM. It can be seen as the brick. That brick is divided up in PE's (or physical extents). These PE's play a major role in data allocation when functionality like striping, mirroring, clustering etc are involved. The PE administration is located in the metadata of the PV. (I know, you need to brush up on the acronyms here). That metadata is kept in an area right after the PV label as well as at the end of the physical volume by default. You can have more, up to 3, copies of the metadata. I would seriously advise you to create at least 2 copies.

```
opensuse:~ # pvdisplay /dev/sdb
  --- Physical volume ---
  PV Name               /dev/sdb
  VG Name               LST0
  PV Size               8.00 GiB / not usable 4.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              2047
  Free PE               255
  Allocated PE          1792
  PV UUID               2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW
```

In this case the PV has 2047 Physical extents of which 255 are free and 1792 are allocated already. The size of each PE is 4 MB which is the atomic entity for data storage.

Depending on the type of logical volume you create these PE's will then be addressed accordingly.

Damaging the label and metadata on a PV is quite annoying but not 100% destructive. LVM keeps it's configuration data in `/etc/lvm/archive` and `/etc/lvm/backup`. Whenever a change is made the previous configuration is kept in the `/etc/lvm/backup` directory which you can use to recover the PV and VG.

Both the backup and archive files are ASCII so can be easily checked.

6.4 VG - Volume Groups

The volume groups are the pools of capacity derived from the capacity of the PV's in those groups. The VG provides a representation of the PE's that are provided by the PV's and will be provisioned to the LV's as Logical Extents (LE).

```
opensuse:~ # vgdisplay LST0 -v
  --- Volume group ---
  VG Name               LST0
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  12
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                1
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               15.99 GiB
  PE Size               4.00 MiB
  Total PE              4094
  Alloc PE / Size       514 / 2.01 GiB
  Free  PE / Size       3580 / 13.98 GiB
  VG UUID               YwWk0A-o1ba-mHSs-od7G-Mir2-YPWP-axHIPa

  --- Logical volume ---
  LV Path                /dev/LST0/lvol0
  LV Name                lvol0
  VG Name                LST0
  LV UUID                132jbo-csDk-brKk-Nx9c-1NHG-7h6z-HRWYT6
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-28 17:14:36 +1000
  LV Status              available
  # open                 0
  LV Size                1.00 GiB
  Current LE             256
  Mirrored volumes       2
```

```
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:7

  --- Physical volumes ---
  PV Name                /dev/sdb
  PV UUID                2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW
  PV Status              allocatable
  Total PE / Free PE     2047 / 1790

  PV Name                /dev/sdc
  PV UUID                Te1V4F-geOv-viVQ-6owe-OFrT-0hbJ-f5YFmO
  PV Status              allocatable
  Total PE / Free PE     2047 / 1790
```

6.5 LV - Logical Volumes

The LV is the highest level of the LVM stack. This layer will be representing the collection of Physical Extents as Logical Extents to the operating system and can be used by the layers we've discussed before.

A few things to take into account.

- Do not further partition a logical volume.
- Allocate what you need now, not next year.
- Ensure the correct volume type is selected. (Raid, Striped, Linear)
- Check if thin provisioning really needed. A thin provisioned volume may be useful in some circumstances but requires a rigorous monitoring regime.

Thin provisioned volumes are pulled out of a special pool called – tadaaa – `thin-pool`. The `thin-pool` keeps track of allocation and mapping of PE to LE and can also de-allocate a LE when it is no longer being used therefore freeing up space in the `thin-pool`.

6.6 Thin Volumes

The example below shows what you need to take into account. A thin-pool is created and takes the default name of a logical volume. This is however not created via the device-mapper as it is some sort of pseudo device.

```
opensuse:~ # lvcreate --type thin-pool -L 8G LST0
  Thin pool volume with chunk size 64.00 KiB can address at most 15.81 TiB of data.
```

```
  Logical volume "lvol1" created.
opensuse:~ # lvdisplay
  --- Logical volume ---
  LV Name                lvol1
  VG Name                LST0
  LV UUID                yOOo22-xGpd-Nzhf-MONe-eEia-jx9T-FuxvF0
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-29 16:04:02 +1000
  LV Pool metadata       lvol1_tmeta
  LV Pool data           lvol1_tdata
  LV Status              available
  # open                 0
  LV Size                8.00 GiB
  Allocated pool data    0.00%
  Allocated metadata     10.79%
  Current LE             2048
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:2
```

As you can see in the LV Name parameter there is no reference to a `/dev/xxxxx/xxx`. From there we can now create a thin provisioned logical volume.

```
opensuse:~ # lvcreate -T -V 15G --thinpool lvol1 LST0
  WARNING: Sum of all thin volume sizes (15.00 GiB) exceeds the size of thin pool LST0/lvol1
  and the amount of free space in volume group (7.98 GiB).
  WARNING: You have not turned on protection against thin pools running out of space.
  WARNING: Set activation/thin_pool_autoextend_threshold below 100 to trigger automatic
  extension of thin pools before they get full.
  Logical volume "lvol2" created.
```

Whoaaaa, whats this warning?? As you can see the thin-pool size was configured with a size of 8GB whilst now we're creating a logical volume of 15GB. As the allocation of PE's is done dynamically upon use there is no problem during creation but it may impose a capacity shortage when the volume gets used. There is a provision in LVM that can keep track of the allocation of extents out of the thin-pool and automatically grow that thin-pool if the utilisation sits above a certain percentage. As a best practise ensure that there is enough free space in the VG's and configure the LVM global configuration to automatically extend the space of the thin-pool by 20% when 80% utilisation has been reached. This can be done by adjusting two values in the `/etc/lvm/lvm.conf`.

- `thin_pool_autoextend_threshold = 80`

- thin_pool_autoextend_percent = 20

When you've set these restart the lvm-monitor service (or restart the host). Check with the `lvmconfig` command if the settings are active.

```
opensuse:~ # lvmconfig
config {
 checks=1
 abort_on_errors=0
 profile_dir="/etc/lvm/profile"
}
<snip>
activation {
 raid_fault_policy="warn"
 mirror_image_fault_policy="remove"
 mirror_log_fault_policy="allocate"
 snapshot_autoextend_threshold=80
 snapshot_autoextend_percent=20
 thin_pool_autoextend_threshold=80  # <<<<<
 thin_pool_autoextend_percent=20    # <<<<<
 monitoring=1
 polling_interval=15
 activation_mode="degraded"
}
```

You can see the association of the two volumes with the `lvdisplay` command.

```
opensuse:~ # lvdisplay
  --- Logical volume ---
  LV Name                lvol1
  VG Name                LST0
  LV UUID                yOOo22-xGpd-Nzhf-MONe-eEia-jx9T-FuxvF0
  LV Write Access        read/write (activated read only)
  LV Creation host, time opensuse, 2020-10-29 16:04:02 +1000
  LV Pool metadata       lvol1_tmeta
  LV Pool data           lvol1_tdata
  LV Status              available
  # open                 2
  LV Size                8.00 GiB
  Allocated pool data    0.00%
  Allocated metadata     10.84%
  Current LE             2048
  Segments               1
  Allocation             inherit
```

```
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:2

  --- Logical volume ---
  LV Path                /dev/LST0/lvol2
  LV Name                lvol2
  VG Name                LST0
  LV UUID                kF3hVC-AKc6-fZmC-88FN-HnyN-0k3D-EClhf3
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-29 16:10:41 +1000
  LV Pool name           lvol1
  LV Status              available
  # open                 0
  LV Size                15.00 GiB
  Mapped size            0.00%
  Current LE             3840
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:4
```

As shown above the pool volume is 8GB and the logical volume `lvol2` is virtually assigned 15GB (or 3840 LE's). The `lvol1` volume is set as read/write with read-only activated. This basically means that only the logical volumes created from this thin-pool can actually read and write and there is no further reference in the OS for use by other tools or applications. As `lvol2` is currently does not have any data yet, the `Mapped size` value is still 0%.

Obviously when we want to use `lvol2` we need to format it with a filesystem and mount it.

```
opensuse:~ # mkfs.ext4 /dev/LST0/lvol2
mke2fs 1.45.6 (20-Mar-2020)
Discarding device blocks: done
Creating filesystem with 3932160 4k blocks and 983040 inodes
Filesystem UUID: 375b814a-714f-4ff3-adc2-4b068b063e3c
Superblock backups stored on blocks:
  32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
opensuse:~ # mkdir /mnt/thinvol

opensuse:~ # mount /dev/LST0/lvol2 /mnt/thinvol/

opensuse:~ # cd /mnt/thinvol/

opensuse:/mnt/thinvol # dd if=/dev/urandom of=example.rnd bs=1024 count=320
320+0 records in
320+0 records out
327680 bytes (328 kB, 320 KiB) copied, 0.0495701 s, 6.6 MB/s

opensuse:/mnt/thinvol # ll -h
total 336K
-rw-r--r-- 1 root root 320K Oct 29 16:17 example.rnd
drwx------ 2 root root  16K Oct 29 16:15 lost+found
```

Having formatted the volume with a `ext4` filesystem and putting a 320KB file on it should show some change in utilisation rate.

```
opensuse:/mnt/thinvol # lvdisplay
  --- Logical volume ---
  LV Name                lvol1
  VG Name                LST0
  LV UUID                yOOo22-xGpd-Nzhf-MONe-eEia-jx9T-FuxvF0
  LV Write Access        read/write (activated read only)
  LV Creation host, time opensuse, 2020-10-29 16:04:02 +1000
  LV Pool metadata       lvol1_tmeta
  LV Pool data           lvol1_tdata
  LV Status              available
  # open                 2
  LV Size                8.00 GiB
  Allocated pool data    3.78%
  Allocated metadata     12.70%
  Current LE             2048
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:2

  --- Logical volume ---
  LV Path                /dev/LST0/lvol2
  LV Name                lvol2
```

```
  VG Name                LST0
  LV UUID                kF3hVC-AKc6-fZmC-88FN-HnyN-0k3D-EClhf3
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-29 16:10:41 +1000
  LV Pool name           lvol1
  LV Status              available
  # open                 1
  LV Size                15.00 GiB
  Mapped size            2.02%
  Current LE             3840
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:4
```

As you can see both the pool and lvol utilisation rates have changed.

6.6.1 Pool threshold condition reached

So what happens if that pool threshold condition of 80% we configured earlier is reached?

I created a couple of dummy files to reach that 80% threshold.

```
opensuse:/mnt/thinvol # ll
total 5632360
-rw-r--r-- 1 root root     327680 Oct 29 16:17 example.rnd
drwx------ 2 root root      16384 Oct 29 16:15 lost+found
-rw-r--r-- 1 root root 1048576000 Oct 29 17:33 random.rnd
-rw-r--r-- 1 root root 1048576000 Oct 29 17:34 random.rnd1
-rw-r--r-- 1 root root 1048576000 Oct 29 17:35 random.rnd2
-rw-r--r-- 1 root root 1048576000 Oct 29 17:37 random.rnd4
-rw-r--r-- 1 root root 1048576000 Oct 29 17:38 random.rnd5
-rw-r--r-- 1 root root  524288000 Oct 29 17:39 random.rnd6
```

Before the threshold was reached we could see that the size of the `this-pool` was 8G but doing the maths on the above we only come to +- 5.4GB. Easily checked with `du`.

```
opensuse:/mnt/thinvol # du -h .
16K ./lost+found
5.4G .
```

And yet the `thin-pool` still expanded with 1.6GB from 8GB to 9.6GB (20% of 8GB)

```
opensuse:/mnt/thinvol # lvdisplay
  --- Logical volume ---
```

```
  LV Name                lvol1
  VG Name                LST0
  LV UUID                yOOo22-xGpd-Nzhf-MONe-eEia-jx9T-FuxvF0
  LV Write Access        read/write (activated read only)
  LV Creation host, time opensuse, 2020-10-29 16:04:02 +1000
  LV Pool metadata       lvol1_tmeta
  LV Pool data           lvol1_tdata
  LV Status              available
  # open                 2
  LV Size                9.60 GiB
  Allocated pool data    69.26%
  Allocated metadata     37.73%
  Current LE             2458
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:2

  --- Logical volume ---
  LV Path                /dev/LST0/lvol2
  LV Name                lvol2
  VG Name                LST0
  LV UUID                kF3hVC-AKc6-fZmC-88FN-HnyN-0k3D-EClhf3
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-29 16:10:41 +1000
  LV Pool name           lvol1
  LV Status              available
  # open                 1
  LV Size                15.00 GiB
  Mapped size            44.33%
  Current LE             3840
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:4
```

So how is that possible? Remember that the atomic entity of a logical volume is the Logical Extend which is then mapped to a Physical Extent on a PV. The way I created the random.xxxx files was with `dd` using 1 MB block sizes times the count required to reach that 1GB and 500MB file. Now be aware that using the `1M` parameter with `dd` is not 2^10 but more 1000*1000 in bytes. These do not align nicely with the 4MB PE size and as such the

usage of the LE's and PE's are somewhat different that you expect.

This also a much misunderstood concept when it comes to thin provisioning on storage arrays. The underlying allocation unit is paramount to the concepts of the thin provisioning algorithms.

6.6.2 File removal

What happens when one or more of these large files are deleted? That depends a bit on the filesystem that resides on it and how it will inform the LVM layer of the deletion.

The mount options of most filesystems have a `-o discard` option. This tells the filesystem to notify the underlying layers that certain blocks are no longer in use and can be replenished into the thin provisioned pool (if any). It is not always a good idea to have this mount option specified as it may incur some performance problems if many write-commands are used who both create new files as well as delete old files. The file-deletion will not only remove the file-system entries but also needs to wait for the underlying layer to complete these discards.

The other option is to schedule a `fstrim` command to the file-system on a controlled time. This can be easily achieved by tailing any other job that may have completed or just on a fixed time by using cron or via the maintenance scheduling facility from Ansible, Puppet etc....

Below is a comparison of what happens when files are deleted on volumes mounted with the `-o discard` option set or not. The layout is there are two VG's with each having a single thin-pool configured out of which two identical logical volumes are carved. LV `V0L2` and `V0L3` have an `XFS` filesystem of which one is mount with the `-o discard` option. The same on the other VG but the only difference is that these two LV's have `EXT4` formatted. Each of these are mounted under /mnt and they have 6 * 1GB (1000*1000) files created.

```
opensuse:~ # lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  V0L2 LST0 Vwi-aotz-- 16.00g tp0         34.99
  V0L3 LST0 Vwi-aotz-- 16.00g tp0         34.99
  tp0  LST0 twi-aotz-- 13.83g             80.98  38.12
  V1L2 LST1 Vwi-aotz-- 16.00g tp1         37.30
  V1L3 LST1 Vwi-aotz-- 16.00g tp1         37.30
  tp1  LST1 twi-aotz-- 13.83g             86.32  40.57

opensuse:~ # mount
<snip>
/dev/mapper/LST0-V0L2 on /mnt/V0L2 type xfs (rw,relatime,attr2,inode64,logbufs=8, \
  logbsize=64k, sunit=128,swidth=128,noquota)
/dev/mapper/LST0-V0L3 on /mnt/V0L3 type xfs (rw,relatime,attr2,discard,inode64, \
```

```
  logbufs=8, logbsize=64k,sunit=128,swidth=128,noquota)
/dev/mapper/LST1-V1L2 on /mnt/V1L2 type ext4 (rw,relatime,stripe=16)
/dev/mapper/LST1-V1L3 on /mnt/V1L3 type ext4 (rw,relatime,discard,stripe=16)

opensuse:~ # ls /mnt/*
/mnt/V0L2:
random.rnd1  random.rnd2  random.rnd3  random.rnd4  random.rnd5  random.rnd6

/mnt/V0L3:
random.rnd1  random.rnd2  random.rnd3  random.rnd4  random.rnd5  random.rnd6

/mnt/V1L2:
lost+found  random.rnd1  random.rnd2  random.rnd3  random.rnd4  random.rnd5  random.rnd6

/mnt/V1L3:
lost+found  random.rnd1  random.rnd2  random.rnd3  random.rnd4  random.rnd5  random.rnd6
```

The utilisation of the volumes as shown above may suprise you somewhat. As I mentioned the volumes have been formated with a different filesystem but for the purpose of the file removal example this doesn't really matter.

When one file of each volume is removed you will see a difference immediatly

```
opensuse:~ # rm /mnt/*/random.rnd6
opensuse:~ # ls /mnt/*
/mnt/V0L2:
random.rnd1  random.rnd2  random.rnd3  random.rnd4  random.rnd5

/mnt/V0L3:
random.rnd1  random.rnd2  random.rnd3  random.rnd4  random.rnd5

/mnt/V1L2:
lost+found  random.rnd1  random.rnd2  random.rnd3  random.rnd4  random.rnd5

/mnt/V1L3:
lost+found  random.rnd1  random.rnd2  random.rnd3  random.rnd4  random.rnd5
opensuse:~ # lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  V0L2 LST0 Vwi-aotz-- 16.00g tp0         34.99
  V0L3 LST0 Vwi-aotz-- 16.00g tp0         29.17
  tp0  LST0 twi-aotz-- 13.83g             74.24  35.84
  V1L2 LST1 Vwi-aotz-- 16.00g tp1         37.30
  V1L3 LST1 Vwi-aotz-- 16.00g tp1         31.48
  tp1  LST1 twi-aotz-- 13.83g             79.59  38.24
```

As you can see the two volumes that had the `-o discard` flag set have immediately returned the LE's to their respective pools. As mentioned the two volumes, V0L2 and V1L2 have not returned their LE's and still hold on to them. When the `fstrim` command is executed against them you will see that the filesystem will inform the logical volume that these blocks are now no longer used and they can be replenished back into the pool.

```
opensuse:~ # fstrim /mnt/V0L2
opensuse:~ # lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  V0L2 LST0 Vwi-aotz-- 16.00g tp0         29.17
  V0L3 LST0 Vwi-aotz-- 16.00g tp0         29.17
  tp0  LST0 twi-aotz-- 13.83g             67.51  33.57
  V1L2 LST1 Vwi-aotz-- 16.00g tp1         37.30
  V1L3 LST1 Vwi-aotz-- 16.00g tp1         31.48
  tp1  LST1 twi-aotz-- 13.83g             79.59  38.24
```

The above output is as expected. `L0V2` is now equally using the allocated LE's as `V0L3`. It requires a rigorous administrative regime to keep on top of file-system allocation, logical volume usage and pool space.

If the `errorwhenfull` option is disabled (default) the write IO's will be queued for the amount of seconds set by the `no_space_timeout` parameter of the `dm_thin_pool` kernel module, this is 60 seconds by default. This may be an option if you have spare PV's available on the system and a monitoring methodology to keep track of the use of the thin_pool LE's. As soon as these are exhausted that monitoring tool may kick of an action to automatically add one or more PV's in that VG.

If you do not have such an option, i.e. no spare PV's or no monitoring option, to some extend the write IO's may be journalled to via the filesystem but in most cases this will result in data corruption.

To prevent such a scenario to occur ensure that

- A PV is added immediatly into the VG, therefore increasing the number of PE's, these can then be used right away or
- Change the `errorwhenfull` parameter so that errors are reported immediately to the file-system and/or application layer.

The latter may result in a crash of the application but datacorruption itself is less likely to occur on a journalled filesystem. The application would still be doing it's internal checks to see what happend and if it needs to correct anything.

In the example below the `tp0` thin-pool has exhausted its space.

```
opensuse:~ # lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  V0L2 LST0 Vwi-aotz-- 16.00g tp0         53.89
```

```
  V0L3 LST0 Vwi-aotz-- 16.00g tp0         29.17
  tp0  LST0 twi-aotz-- 13.83g             98.79  43.57
  V1L2 LST1 Vwi-aotz-- 16.00g tp1         37.30
  V1L3 LST1 Vwi-aotz-- 16.00g tp1         31.48
  tp1  LST1 twi-aotz-- 13.83g             79.59  38.24

opensuse:~ # lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  V0L2 LST0 Vwi-aotz-- 16.00g tp0         56.54
  V0L3 LST0 Vwi-aotz-- 16.00g tp0         29.17
  tp0  LST0 twi-aotzD- 13.83g             100.00 44.55
  V1L2 LST1 Vwi-aotz-- 16.00g tp1         37.30
  V1L3 LST1 Vwi-aotz-- 16.00g tp1         31.48
  tp1  LST1 twi-aotz-- 13.83g             79.59  38.24
```

The writes to the file-system on `V0L2` and `V0L3` are still possible as LVM is not directly returning the error. This happens later. As you can see from the configuration below the `errorwhenfull` parameter is disabled.

```
opensuse:~ # lvs -o lv_full_name,lv_health_status,lv_when_full
  LV        Health          WhenFull
  LST0/V0L2
  LST0/V0L3
  LST0/tp0  out_of_data     queue
  LST1/V1L2
  LST1/V1L3
  LST1/tp1                  queue
```

After the 60 seconds queue-time LVM will log the error via the device-mapper kernel module.

```
Nov 03 10:03:35 opensuse kernel: device-mapper: thin: 254:6: reached low
  water mark for data  device: sending event.
Nov 03 10:03:35 opensuse systemd[1]: Starting Cleanup of Temporary Directories...
Nov 03 10:03:39 opensuse lvm[922]: Insufficient free space: 708 extents
  needed, but only 544 available
Nov 03 10:03:39 opensuse lvm[922]: Failed command for LST0-tp0-tpool.
Nov 03 10:03:39 opensuse lvm[922]: WARNING: Thin pool LST0-tp0-tpool data
  is now 95.68% full.
Nov 03 10:03:48 opensuse systemd[1]: systemd-tmpfiles-clean.service: Succeeded.
Nov 03 10:03:48 opensuse systemd[1]: Finished Cleanup of Temporary Directories.
Nov 03 10:04:29 opensuse kernel: device-mapper: thin: 254:6: switching pool to
  out-of-data-space (queue IO) mode
Nov 03 10:04:29 opensuse lvm[922]: WARNING: Thin pool LST0-tp0-tpool data is now
  100.00% full.
```

```
Nov 03 10:04:29 opensuse lvm[922]: Insufficient free space: 885 extents needed,
  but only 544 available
Nov 03 10:04:29 opensuse lvm[922]: Failed command for LST0-tp0-tpool.
Nov 03 10:05:30 opensuse kernel: device-mapper: thin: 254:6: switching pool to
  out-of-data-space (error IO) mode
Nov 03 10:05:30 opensuse kernel: dm-9: writeback error on inode 140, offset 620756992,
  sector 21143680
Nov 03 10:05:30 opensuse kernel: dm-9: writeback error on inode 140, offset 624951296,
  sector 21151872
Nov 03 10:05:30 opensuse kernel: dm-9: writeback error on inode 140, offset 629145600,
  sector 21160064
Nov 03 10:05:30 opensuse kernel: dm-9: writeback error on inode 140, offset 633339904,
  sector 21168256
Nov 03 10:05:30 opensuse kernel: dm-9: writeback error on inode 140, offset 637534208,
  sector 21176448
Nov 03 10:05:30 opensuse kernel: dm-9: writeback error on inode 140, offset 641728512,
  sector 21184640
----- etc etc etc
```

Obviously this is not what you want. The behaviour when the `errorwhenfull` parameter is set on the thin-pool is much different. First thing to do is to ensure operations can resume on the LV's. As mentioned, removing files from a filesystem that has the `-o discard` option mount or by using the `fstrim` command will help you get back on track. If you cannot delete data you may be able to move it to a different location or add PV's

After removing some files and using `fstrim` on `V0L2` the state of the thin-pool `tp0` looks good again.

```
opensuse:/mnt/V0L3 # lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  V0L2 LST0 Vwi-aotz-- 16.00g tp0         33.97
  V0L3 LST0 Vwi-aotz-- 16.00g tp0         29.17
  tp0  LST0 twi-aotz-- 13.83g             73.88  35.47
  V1L2 LST1 Vwi-aotz-- 16.00g tp1         37.30
  V1L3 LST1 Vwi-aotz-- 16.00g tp1         31.48
  tp1  LST1 twi-aotz-- 13.83g             79.59  38.24
```

By now changing the error parameter the result of a full pool will be different

```
opensuse:/mnt/V0L2 # opensuse:/mnt/V0L2 # lvchange --errorwhenfull y /dev/LST1/tp1
  Logical volume LST1/tp1 changed.

opensuse:/mnt/V0L2 # lvs -o lv_full_name,lv_health_status,lv_when_full
  LV        Health          WhenFull
```

```
  LST0/V0L2
  LST0/V0L3
  LST0/tp0  out_of_data     error
  LST1/V1L2
  LST1/V1L3
  LST1/tp1                  error
```

Filling up the volume `V1L2` shows the `tp1` pool becoming exhausted. The EXT4 filesystem on `V1L2` was mounted with the `-o errors=remount-ro` and given that the underlying LVM reported errors back to the file-system the response was as expected.

```
opensuse:~ # lvs
  LV   VG   Attr       LSize  Pool Origin Data%  Meta%  Move Log Cpy%Sync Convert
  V0L2 LST0 Vwi-aotz-- 16.00g tp0         0.07
  V0L3 LST0 Vwi-aotz-- 16.00g tp0         0.07
  tp0  LST0 twi-aotz--  8.00g             0.27   10.89
  V1L2 LST1 Vwi-aotz-- 16.00g tp1         2.38
  V1L3 LST1 Vwi-aotz-- 16.00g tp1         84.05
  tp1  LST1 twi-aotzD- 13.83g             100.00 43.44

opensuse:/mnt/V1L3 # dd if=/dev/urandom of=disk-full.rnd${i} count=1000000 bs=1024
<snip>
1024000000 bytes (1.0 GB, 977 MiB) copied, 29.4122 s, 34.8 MB/s
1000000+0 records in
1000000+0 records out
1024000000 bytes (1.0 GB, 977 MiB) copied, 22.0816 s, 46.4 MB/s
dd: error writing 'disk-full.rnd15': Read-only file system
285591+0 records in
285590+0 records out
292444160 bytes (292 MB, 279 MiB) copied, 65.4181 s, 4.5 MB/s
dd: failed to open 'disk-full.rnd16': Read-only file system
dd: failed to open 'disk-full.rnd17': Read-only file system
dd: failed to open 'disk-full.rnd18': Read-only file system
```

And obviously this will also be reflected by the `mount` output where you will see the `ro` flag active.

```
"/dev/mapper/LST1-V1L3 on /mnt/V1L3 type ext4 (ro,relatime,discard,errors=remount-
ro,stripe=16)"
```

Different file-systems have different mount options and therefore different possibilities. Familiarise yourself with them so you can make an informed decision.

6.7 Cache volumes

I know, I know. I should not touch on performance but this is a feature which may result in grievous bodily harm if done incorrect. :-)

The first thing you need to decide on in the decision of the device mapper caching module. There are two `dm-cache` and `dm-writecache`.

6.7.1 dm-writecache

The `dm-writecache` module is a somewhat simplified caching module which only catches writes to a LV and writes it to the faster device. This is then subsequently being moved from that faster device to the slower-device. There is no movements afterwards from the slower to faster device. All data that is read will be done from the slower device and the data will be stored in the systems page-cache. The most useful target for this behaviour is bursty write intensive applications. The size of the fast-device needs to be large enough to hold these write-burst and not being hampered by the offload to the slower device. You can imaging that if you have a "small" fast device (lets say 500G) but you application is starting to burst 1TB of data that fast device will at some point need to destage the data to that slower volume. This inevitably will therefore impose a contention between the active application writes and the destage data movement and thus can cause application performance come to a grinding halt.

There are a few parameters that you can use to optimise the behaviour. If you have applications that can utilise the `flush` IO command the `dm-writecache` module can immediately flush the configured number of blocks out of the fast to the slow device. This allows a better alignment between the applications behaviour with the `dm-cache` possibilities. If an application has written 200MB but the `dm-writecache` module is configured to de-stage only after having received 300MB you then rely on the default or configured `autocommit_time` for that destage to kick in. That may therefore incur a delay from the default 1 second to 100s of milliseconds or more. If the application had been able to issue the `flush` after the 200MB was written that time, and therefore valuable space on the cache device, could've been saved. It make therefore sense to find out what your applications' options are and how it behaves in such an environment.

When creating a write-cache enabled logical volume some sort of 'ghost' volume is created that takes on the characteristics from the original "slow" volume but is in fact representing the combined "fast" AND "slow" volume. That logic is inserted into the IO path and the `dm-writecache` module does it's magic.

An example of creating such an instance is shown below"

```
vgcreate LST0 /dev/sdb /dev/sdc /dev/sdd /dev/sde
 Volume group "LST0" successfully created
```

```
lvcreate -n fast -L 500M LST0 /dev/sdb
Logical volume "fast" created.

lvcreate -n slow1 -L 1G LST0
Logical volume "slow1" created.

lvchange -a n LST0/fast
lvconvert --type writecache --cachevol fast LST0
Logical volume LST0/slow1 now has writecache.
```

You now have a "slow1" volume which you can put a filesystem on and mount it

```
opensuse:~ # lvs
  LV    VG   Attr       LSize Pool        Origin         Data%  Meta%....
  slow1 LST0 Cwi-a-C--- 1.00g [fast_cvol] [slow1_wcorig] 0.00


opensuse:~ # mkfs.xfs /dev/mapper/LST0-slow1
<snip>
log      =internal log           bsize=4096   blocks=2560, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0


opensuse:~ # mount /dev/mapper/LST0-slow1 /mnt/dmw/

opensuse:~ # mount
<snip>
/dev/mapper/LST0-slow1 on /mnt/dmw type xfs (rw,relatime,attr2,inode64 \
  ,logbufs=8,logbsize=32k,noquota)


opensuse:~ # lvs
  LV    VG   Attr       LSize Pool        Origin         Data%  Meta%....
  slow1 LST0 Cwi-aoC--- 1.00g [fast_cvol] [slow1_wcorig] 2.13
```

The LV attributes already show you there is something special about this device and by adding the -a parameter it shows what

```
opensuse:/mnt/dmw # lvs -a -o lv_name,vg_name,pool_lv,origin,lv_attr,lv_role
  LV             VG   Pool        Origin         Attr       Role
  [fast_cvol]    LST0                           Cwi-aoC--- private
  slow1          LST0 [fast_cvol] [slow1_wcorig] Cwi-aoC--- public
```

```
[slow1_wcorig] LST0                                owi-aoC--- private,writecache\
,origin,writecacheorigin
```

As you can see the "slow1" device is basically a mapping device pointing the to "hidden" devices `[fast_cvol]` and `[slow1_wcorig]`. These are now statically linked. You cannot create a new LV and also link this to the same 'fast' cache device. You would need to create a separate LV cache volume and attach this to a new LV "slow" volume.

 Be aware that if you add internal caching volumes to a disk or set of disks provisioned out of a SAN environment, where errors are seen in that SAN environment, some very unpredictable results may occur.

6.7.2 dm-cache

The more elaborate, or should I say targeted a bit differently, `dm-cache` module has added functionality where it keeps track of which parts of the volume are most accessed and therefore are most likely to benefit from residing on faster storage. The `dm-cache` module also allows to create a cache-pool which splits the data from the meta-data. This may in various circumstances be beneficial for performance reasons large and small IO's are less likely to interfere with each other.

`dm-cache` can operate in three modes 'pass-trough', 'write-back' and 'write-through'. Each mode has it's particular use but it is very depending on application behaviour.

`dm-cache` provides the option of using policies via plug-ins which can adjust the operations of IO's and promotion/demotion of data. The two standard policies that are now delivered are `mq` and `smq` [22]. At the time of this writing mq is aliased to `smq`mainly for the reason that `smq` is much more memory optimised in addition of being handling changing sectors. It does not use a change sector counter but utilises a `hotspot-queue`.

6.7.3 Benefits of caching

This may seem like an kicking in an open door as keeping most often used data close to the application will result in better performance. A second, and often overlooked, scenario is that by using local persistent-memory or SSD's drives as transparent caching modules, the impact on SAN environments can be drastically reduced. By using fast local SSD's in an LVM cached setup in conjunction with SAN provisioned volumes causes the effect that a write intensive, bursty IO-pattern will be "smoothed" by the local cache. Adjusting cache-settings like high and low watermark parameters as well as block-sizes can significantly improve IO and application behaviour in SAN's.

Furthermore, be aware that the file-system and application layers have no knowledge of the underlying LVM setup and cache behaviour. If you need to move applications between storage arrays or the LVM setup is changing be aware that the IO profile that was optimised

as a result of this caching methodology is changing significantly and traffic patterns in SAN's will therefore also change. This may cause a severe impact resulting in a behaviour that requires some significant expertise of SAN and array capabilities to troubleshoot and resolve it.

You will need to do some testing before taking this into production. There is no silver bullet or getting a one-size-fits-all performance optimised caching setup.

6.7.4 Drawbacks of caching and caching errors

There are a lot of knobs and switches that adjust how movement of data between cache and persistent volumes is handled. Movement of data between volumes incurs the use of bandwidth which may have a negative impact on other data transfer operations.

As an example when an application is issuing a lot of reads to certain sectors of a volume the caching algorithm may decide to promote these sectors to the cache volume. If subsequent random read IO's on different sectors on that volume is submitted there will be some sort of contention between the cache movement and application IO. To what extent this will happen is extremely difficult, if not impossible, to predict. It is therefore of the utmost importance to know the application behaviour and any given moment in time and adjust the caching profiles accordingly. Especially in large SAN environments the behaviour of cache volumes in hosts may lead to strange phenomenons. The art of systems-administration comes into play here.

Cache Coherency[1] is another focus point when using caching volumes. Discrepancy of the location of data is one of the primary concerns for developers when write code for any caching system. Irrespective whether this is for cpu, memory, arrays or anything else you can think of if a certain chunk of data is requested by the application it needs to come from the location which has the most recent version of that data.

So what can go wrong here? The main danger is meta-data corruption. If sectors that contain meta-data are corrupted there is a real danger that data corruption is perceived by the operating system or application. Best medicine for that is data-loss prevention and ensure that backups and tested restores are at hand in case things go wrong. Consistency is still a primary concern of the file-system and mirror/raid layers. The `dm-writecache` and `dm-cache` can co-exist with other layers of the device mapper ecosystem.

So much on LVM performance and caching. I know there is a lot more to write on this topic but that will have to wait for a later release.

[1] https://en.wikipedia.org/wiki/Cache_coherence

6.8 LVM corruption scenario

In the below case the label and metadata of the PV information on `/dev/sdb` became corrupt. (what dd can do for ya... :-))

```
opensuse:~ # dd if=/dev/urandom of=/dev/sdb obs=512 seek=1 count=2
2+0 records in
2+0 records out
1024 bytes (1.0 kB, 1.0 KiB) copied, 0.00156159 s, 656 kB/s
```

6.8.1 Damaged PV metadata

```
opensuse:~ # pvdisplay /dev/sdb
  WARNING: Couldn't find device with uuid 2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW.
  WARNING: VG LST0 is missing PV 2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW.
  WARNING: Couldn't find all devices for LV LST0/lvol0 while checking used and \
  assumed devices.
  WARNING: Couldn't find all devices for LV LST0/lvol1 while checking used and \
  assumed devices.
  Failed to find physical volume "/dev/sdb".
```

The UUID is the one thing that is interesting here as that will be needed to correlate the entry in the archive or backup file and be able to restore the label and metadata.

As we did not make an error with the LVM configuration itself but merely shovelled its information from under his its feet we can use the backup file to recover from this.

The backup file in `/etc/lvm/backup/LST0` contains the following information:

```
# Generated by LVM2 version 2.03.05(2) (2019-06-15): Wed Oct 28 14:24:22 2020

contents = "Text Format Volume Group"
version = 1

description = "Created *after* executing 'pvchange --addtag=DEV_B /dev/sdb'"

creation_host = "opensuse"      # Linux opensuse 5.7.11-1-default #1 \
  SMP Wed Jul 29 09:32:21 UTC 2020 (5015994) x86_64
creation_time = 1603859062      # Wed Oct 28 14:24:22 2020

LST0 {
        id = "YwWk0A-o1ba-mHSs-od7G-Mir2-YPWP-axHIPa"
        seqno = 7
        format = "lvm2"                 # informational
        status = ["RESIZEABLE", "READ", "WRITE"]
```

```
        flags = []
        extent_size = 8192              # 4 Megabytes
        max_lv = 0
        max_pv = 0
        metadata_copies = 0

        physical_volumes {

                pv0 {
                        id = "2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW"
                        device = "/dev/sdb"     # Hint only

                        status = ["ALLOCATABLE"]
                        flags = []
                        tags = ["DEV_B"]
                        dev_size = 16777216     # 8 Gigabytes
                        pe_start = 2048
                        pe_count = 2047 # 7.99609 Gigabytes
                }
<snip>
```

As you can see the UUID of PV0 (2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW) is the one were looking for so this is indeed the file we can use to recover the PV and therefore the volumegroup. A very handy piece of information is the description. It shows the last command being used when this backup was created. The files in the archive directory also contain this (it is basically copied and timestamped out of backup) so there is somewhat of a history trail which is very useful for trying to piece together if/when/why something went wrong.

6.8.2 Recover the PV

The way to do this is to recreate the VG with the backup file. The catch is you cannot do this online as LVM will need to recreate the state of the VG.

```
vgchange --activate n --partial LST0
```

```
opensuse:/etc/lvm/backup # vgchange --activate  n --partial LST0
  PARTIAL MODE. Incomplete logical volumes will be processed.
  WARNING: Couldn't find device with uuid 2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW.
  WARNING: VG LST0 is missing PV 2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW.
  WARNING: Couldn't find all devices for LV LST0/lvol0 while checking used and \
  assumed devices.
  WARNING: Couldn't find all devices for LV LST0/lvol1 while checking used and \
  assumed devices.
```

```
  0 logical volume(s) in volume group "LST0" now active
```

The last line now show that the volume group VG0 has no active volumes anymore. Now for the recovery. It basically is a restore of the entire VG where the selection is limited to the UUID and device you specify.

```
pvcreate --uuid="2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW" --restorefile /etc/lvm/backup/LST0
```

```
opensuse:/etc/lvm/backup # pvcreate --uuid "2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW" \
  --restorefile /etc/lvm/backup/LST0 /dev/sdb

  WARNING: Couldn't find device with uuid 2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW.
  WARNING: Couldn't find device with uuid Te1V4F-geOv-viVQ-6owe-OFrT-0hbJ-f5YFmO.
  WARNING: Couldn't find device with uuid 2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW.
  WARNING: VG LST0 is missing PV 2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW.
  Physical volume "/dev/sdb" successfully created.
```

Checking the result shows:

```
opensuse:/etc/lvm/backup # pvdisplay /dev/sdb
  WARNING: PV /dev/sdb in VG LST0 is missing the used flag in PV header.
  --- Physical volume ---
  PV Name               /dev/sdb
  VG Name               LST0
  PV Size               8.00 GiB / not usable 4.00 MiB
  Allocatable           yes
  PE Size               4.00 MiB
  Total PE              2047
  Free PE               255
  Allocated PE          1792
  PV UUID               2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW
```

And a somewhat verbosed output of vgdisplay shows we're almost back in business.

```
opensuse:/etc/lvm/backup # vgdisplay LST0 -v
  WARNING: PV /dev/sdb in VG LST0 is missing the used flag in PV header.
  --- Volume group ---
  VG Name               LST0
  System ID
  Format                lvm2
  Metadata Areas        1
  Metadata Sequence No  7
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                2
```

```
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               15.99 GiB
  PE Size               4.00 MiB
  Total PE              4094
  Alloc PE / Size       1792 / 7.00 GiB
  Free  PE / Size       2302 / 8.99 GiB
  VG UUID               YwWk0A-o1ba-mHSs-od7G-Mir2-YPWP-axHIPa

  --- Logical volume ---
  LV Path                /dev/LST0/lvol0
  LV Name                lvol0
  VG Name                LST0
  LV UUID                9BOZVH-aAic-FzvR-mZjL-tS55-Sk5n-I3M6F4
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-27 18:14:19 +1000
  LV Status              NOT available
  LV Size                6.00 GiB
  Current LE             1536
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto

  --- Logical volume ---
  LV Path                /dev/LST0/lvol1
  LV Name                lvol1
  VG Name                LST0
  LV UUID                VCNWqW-z9mn-Uioo-H190-6nXj-i32t-CITWJg
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-27 18:15:30 +1000
  LV Status              NOT available
  LV Size                1.00 GiB
  Current LE             256
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto

  --- Physical volumes ---
  PV Name               /dev/sdb
  PV UUID               2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW
```

```
  PV Status              allocatable
  Total PE / Free PE     2047 / 255

  PV Name                /dev/sdc
  PV UUID                Te1V4F-geOv-viVQ-6owe-OFrT-0hbJ-f5YFmO
  PV Status              allocatable
  Total PE / Free PE     2047 / 2047
```

The volumegroup information was actually not read from the PV located on `/dev/sdb` It was still available on other metadata blocks. Only the actual PV data was read from the device `/dev/sdb`. Before the restore it would simply show the same information except the PV name (which defaults to the devicename)

```
<snip>
  --- Physical volumes ---
  PV Name                [unknown]                                <<<<<<<<<<<<<<<<
  PV UUID                2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW
  PV Status              allocatable
  Total PE / Free PE     2047 / 255

  PV Name                /dev/sdc
  PV UUID                Te1V4F-geOv-viVQ-6owe-OFrT-0hbJ-f5YFmO
  PV Status              allocatable
  Total PE / Free PE     2047 / 2047
```

6.8.3 Recover the VG

As mentioned the VG still is not really usable. To complete the restore the VG needs to be recovered as well. In order to do that the `vgcfgrestore LST0` command is used.

```
opensuse:/etc/lvm/backup # vgcfgrestore LST0


  Restored volume group LST0.
opensuse:/etc/lvm/backup # vgdisplay LST0 -v
  --- Volume group ---
  VG Name                LST0
  System ID
  Format                 lvm2
  Metadata Areas         2
  Metadata Sequence No   8
  VG Access              read/write
  VG Status              resizable
  MAX LV                 0
```

```
  Cur LV                2
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               15.99 GiB
  PE Size               4.00 MiB
  Total PE              4094
  Alloc PE / Size       1792 / 7.00 GiB
  Free  PE / Size       2302 / 8.99 GiB
  VG UUID               YwWk0A-o1ba-mHSs-od7G-Mir2-YPWP-axHIPa

  --- Logical volume ---
  LV Path                /dev/LST0/lvol0
  LV Name                lvol0
  VG Name                LST0
  LV UUID                9BOZVH-aAic-FzvR-mZjL-tS55-Sk5n-I3M6F4
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-27 18:14:19 +1000
  LV Status              NOT available
  LV Size                6.00 GiB
  Current LE             1536
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto

  --- Logical volume ---
  LV Path                /dev/LST0/lvol1
  LV Name                lvol1
  VG Name                LST0
  LV UUID                VCNWqW-z9mn-Uioo-H190-6nXj-i32t-CITWJg
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-27 18:15:30 +1000
  LV Status              NOT available
  LV Size                1.00 GiB
  Current LE             256
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto

  --- Physical volumes ---
  PV Name               /dev/sdb
```

```
  PV UUID               2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW
  PV Status             allocatable
  Total PE / Free PE    2047 / 255

  PV Name               /dev/sdc
  PV UUID               Te1V4F-geOv-viVQ-6owe-OFrT-0hbJ-f5YFmO
  PV Status             allocatable
  Total PE / Free PE    2047 / 2047

    Archiving volume group "LST0" metadata (seqno 7).
    Archiving volume group "LST0" metadata (seqno 8).
    Creating volume group backup "/etc/lvm/backup/LST0" (seqno 8).
```

From the above there are two things that you should take note off. The "WARNING" message of the missing "used flag" disappears after the `vgrestore` command and LVM should have created a new backup file and a archive copy in the respective `/etc/lvm/[archive/backup]` locations. Furthermore you see that the logical volumes are still not usable as the "LV Status" is flagged as being not available.

6.8.4 Reactivate the VG

Remember that we deactivated the volumegoup before we started the recovery so we now need to activate it again. To do so use the `vgchange` command again.

```
opensuse:/etc/lvm/backup # vgchange --activate y LST0
  2 logical volume(s) in volume group "LST0" now active
```

And were back in business.

```
opensuse:/etc/lvm/backup # vgdisplay LST0 -v
  --- Volume group ---
  VG Name               LST0
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  8
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                2
  Open LV               0
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               15.99 GiB
```

```
  PE Size               4.00 MiB
  Total PE              4094
  Alloc PE / Size       1792 / 7.00 GiB
  Free  PE / Size       2302 / 8.99 GiB
  VG UUID               YwWk0A-o1ba-mHSs-od7G-Mir2-YPWP-axHIPa

  --- Logical volume ---
  LV Path                /dev/LST0/lvol0
  LV Name                lvol0
  VG Name                LST0
  LV UUID                9BOZVH-aAic-FzvR-mZjL-tS55-Sk5n-I3M6F4
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-27 18:14:19 +1000
  LV Status              available
  # open                 0
  LV Size                6.00 GiB
  Current LE             1536
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:0

  --- Logical volume ---
  LV Path                /dev/LST0/lvol1
  LV Name                lvol1
  VG Name                LST0
  LV UUID                VCNWqW-z9mn-Uioo-H190-6nXj-i32t-CITWJg
  LV Write Access        read/write
  LV Creation host, time opensuse, 2020-10-27 18:15:30 +1000
  LV Status              available
  # open                 0
  LV Size                1.00 GiB
  Current LE             256
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     1024
  Block device           254:1

  --- Physical volumes ---
  PV Name               /dev/sdb
```

```
  PV UUID               2Zlix1-VkoX-HhWx-dELn-bC1X-u6MQ-JTc8CW
  PV Status             allocatable
  Total PE / Free PE    2047 / 255

  PV Name               /dev/sdc
  PV UUID               Te1V4F-geOv-viVQ-6owe-OFrT-0hbJ-f5YFmO
  PV Status             allocatable
  Total PE / Free PE    2047 / 2047
```

6.9 Meta-data

The internal operations and maintenance task of LVM handles a lot of meta-data and is paramount for proper operations of LVM. Especially when it comes to advanced features like thin-provisioning, CoW (Copy-on-Write), snap-shots etc. The allocation of the meta data is by default on the first data disk of the VG. Separate metadata disks can be assigned when the LV is created and there may be good reasons for this. You may need to ensure that the metadata is located on a more protected volume than the data-disks in case these are simple linear or striped LV's without further protection. The below shows such a setup where the meta-data of thin-pool `tp1` only sits on the `sdd` drive. Losing that `sdd` drive will have consequences in the sense the LV is basically no longer able to reference any of its LE's.

```
LST1-V1L3          ext4                    cb4fe624-bded-4730-b6cd-a8ef1558cab5      \
  6.2G    55% /mnt/V1L3

 -LST1-tp1-tpool
  |-LST1-tp1_tmeta
  | |-sdd          LVM2_member             vzihd1-vrUN-5fkp-BoxZ-4Aia-OJiU-to0WCC
  |-LST1-tp1_tdata
    |-sdd          LVM2_member             vzihd1-vrUN-5fkp-BoxZ-4Aia-OJiU-to0WCC
    |-sde          LVM2_member             sjcBwL-1YBO-8BGz-Pkx9-jjsF-xn0p-FmVxMu
```

6.10 Performance problems

Even though I mentioned in the introduction that performance was not an important point of attention at this stage I want to touch on it over here slightly. Depending on the provisioned hardware and configuration setup, it is very easy to make mistakes that will lead to significant performance problems. As an example the disks I've used in the scenario are all located on the same physical disk in a Virtual Machine. A similar problem can easily be recreated when two or more partitions on the same disks are used for PV's and subsequently aggregated in a single VG or being used in different VG's but the logical volume that is carved out of the VG's is set to be a mirror. When data growth requires more space you'll see that more PV's will be added to the volumegroup and logical volumes will be extended.

As per your documentation and skills as a sysadmin you may be 100% across optimal configurations and settings but don't be surprised that configurations such as below may sometimes land on your desk.

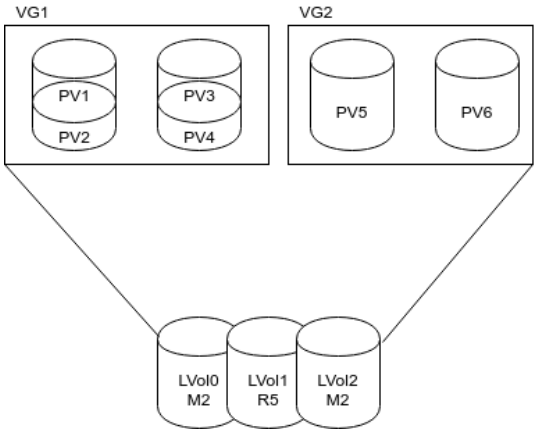


Figure 6.1: Incorrect LVM layout

You'll see that two disks have actual partitions on them and these are set to be in VG1 whereas two other disks are allocated in VG2. Out of these there are three logical volumes created with each different storage characteristics. Lvol0 is a RAID1 volume with two mirror copies, Lvol1 is a RAID5 and Lvol2 is also a mirrored volume with two copies. Technically there is no-one preventing you from doing this. From a performance perspective this is creating nightmares and different workload profiles will have a severe adverse impact on all applications that rely on this.

What needs to be done is first of all ensure that workload profiles based on application behaviour is clear. Then the hardware needs to be lined up to support these workloads subsequently followed by an optimal configuration in LVM. Rule of thumb is that you will ALWAYS have a disadvantage compared to dedicated storage hardware when it comes to striping and mirroring. Multiple decades of ASIC and FPGA design have ensured that dealing with a multitude of workload profiles, caching methodologies and firmware optimisations will very likely out-perform your LVM setup. Using LVM to provide a wide striping or mirroring option is not likely to improve performance as your CPU's have better things to do than to keep track of IO handling. That being said, if your options are limited either technically or financially LVM may provide you some great options.

From and administrative point-of-view LVM provides a vast toolkit of options moving data around. If you're planning for a tech-refresh where you bought a new set of storage arrays it is a breeze to migrate these over with LVM.

6.10.1 Monitoring performance

LVM2 does not provide a dedicated tool to monitor performance characteristics. The level at which LVM operates, device mapper, can be monitored by the `dmstats` tool. I would highly advise to get familiar with this tool as it allows to hone in on very specific area of the device-mapper, and therefore LVM, structures. Normal tools like `iostats` may not clearly indicate the association between physical and logical devices. The `dmstats` tool provides configuration options to correlate different aspects of the storage structure and device allocation. It even can provide performance characteristics for individual files. This may become very useful when reports and stats need to be analysed when databases or virtual machine images are located on a device-mapper backed volume. This will provide a clearer overview of what is happening in case you run into performance issues. There are some pre-requisites such as the file-system needs to support the FIEMAP ioctl. That will return the physical extents location of the file on that filesystem and thus can then be mapped into a `dmstats` region.

Chapter 7

RAID

RAID stands for `Redundant Array of Independent Disks`. Colloquially also referred as `Redundant Array of Inexpensive Disks`. Basically what is does it distributing chunks of data over two or more disks to either have a full copy or a striped set with parity to be able to overcome one or more physical disk failures. Raid is well known and documented however its application use case is somewhat outdated. The main reason I say this is that the recovery-time of failed physical disks has grown significantly over the last decade or so mainly because the capacity of the physical disks has grown to multiple terabytes. Any failure of one disk will require a read action on the surviving members of the raid group, an XOR parity calculation on each data block (or stripe), and a write action to the spare disk. All this in addition to the existing workload that was already in place before a disk failed. I'm not saying that RAID is bad however it should not be used as a single source of redundancy when you have very business critical systems.

Other mechanisms that cater for a more distributed way of placing data on storage devices should be investigated. As usual this depends on your use case and the capabilities of your application.

7.1 Terminology

When it comes to RAID some things need to be understood first. You will often see the words `stripe`, `stripe unit`, `stripe width`, `chunk`, `strip` and a few more. Basically what do these mean? In order to distribute a certain set of data, it will need to be split up in sections where each section is written to a different disk. The sections themselves are called a `chunks`, `strides` or `stripe units` and the amount of data it holds is called `chunk size`, `stride size`, `stripe size`, `stripe depth` or `stripe length`.
The collection of one set of `chunks` belonging to the same group is called a `strip`. The amount

of chunks that need to be created should be based on the underlying hardware configuration. A raid6 set of 10 data disks and two parity disks has a stripe width of 10 as you do not count the parity disks.

As you've seen in the File System chapter this is important to know the underlying infrastructure in order to be able to create volumes that are able to utilize the hardware capabilities to the maximum.

7.2 MD

Linux has a software based raid-capability called MD. (Which stands for Multiple Device driver). It's a kernel based driver which lets you configure certain *personalities* like RAID1, RAID10, RAID4 and RAID5. RAID1 basically standing for disk mirroring, RAID4 for distributed data with dedicated parity and RAID5 for distributed data with distributed parity 1(I won't go into the technical details on RAID algorithms).

A RAID0 option is also provided which allows you to create a single spanned volume without any redundancy. This is only advisable in a RAID10 (striped RAID1) setup as the chances of losing data are growing exponentially with every disk you add to a RAID0 set.

As I mention in the previous chapter the md driver is also used by LVM so you will see some similarities here.

Be aware that the md driver, although being a low-level kernel mode driver, can have a significant performance impact on IO behaviour. In case of RAID3 and RAID5 data needs to be split into chunks, parity needs to be calculated and all segments then are written to the disks. The same is true when the data needs to be read. A RAID3 or RAID5 set contains at least 3 disks which means that the data that needs to be read incur a read IO to at least these members as well as an IO to the disk where the parity chunk is located. In SCSI based systems a lot can be gained with queuing mechanisms etc but be aware that all these calculations are done by the CPU of the host. If very large RAID sets containing many disks or many raid-sets are handled by the host this will for sure have a negative effect on OS and application performance. A RAID1 set has the benefit of improving the IO read capabilities by a factor of the number of mirror-members. Each disk can serve the same data so the md driver can request data from multiple disks based on certain algorithms like outstanding IO's, requested size etc. (The statement is not 100% true as there are other factors involved but as a rough guideline it will suffice). The data that is written is alway duplicated so from a capacity and cost perspective it is the least efficient. RAID1 does not incur the raid-calculation penalty that RAID3 and RAID5 have however each write IO from the application will need to be written to all members of the mirrorset.

It is recommended to have a dedicated RAID controller or an external RAID array to handle the redundancy and IO requests. These are far better equipped to manage a high IO workload whilst offloading the RAID calculations to dedicated ASIC's (Application Specific

Integrated Chips) therefore freeing up cycles of the host CPU. The benefit of these controllers is also that command-destaging is possible meaning that the host OS is not tasked of keeping track of outstanding IO's to individual drives. The read or write command is basically sent to the raid controller with the expectation that the controller takes care of the rest and only the data and subsequent status message is returned the host. Most modern RAID controllers have drivers with extensive management capabilities so even from an OS perspective you may be able to manage the RAID configuration via the CLI or some sort of web interface. You get what you pay for so properly investigate what you need/want out of a controller.

That being said there is a very good reason to use a software raid solution. If you work from home as I do and have a limited budget to buy hardware it makes perfectly sense to just get a set of relatively cheap hard-disks and put these in a raidset. In case one of them observes a failure you still have the other one to work of. In case you have no local data and save your work on Github (or any of the cloud providers) for backup purposes you may not need it.

An example of a software raid setting:

```
opensuse:~ # mdadm --detail /dev/md1
/dev/md1:
           Version : 1.2
     Creation Time : Tue Jun 30 17:45:22 2020
        Raid Level : raid1
        Array Size : 8379392 (7.99 GiB 8.58 GB)
     Used Dev Size : 8379392 (7.99 GiB 8.58 GB)
      Raid Devices : 2
     Total Devices : 3
       Persistence : Superblock is persistent

       Update Time : Tue Aug  4 13:40:53 2020
             State : clean
    Active Devices : 2
   Working Devices : 3
    Failed Devices : 0
     Spare Devices : 1

Consistency Policy : resync

              Name : opensuse:md1  (local to host opensuse)
              UUID : 8a1c4a76:97b69b4a:51bb6931:5c3d70c7
            Events : 26

    Number   Major   Minor   RaidDevice State
       0       8       32        0      active sync   /dev/sdc
```

```
1       8       48        1      active sync   /dev/sdd

2       8       64        -      spare   /dev/sde
```

As shown in the above example the md driver creates a logical device called md1 and p1 is its first partition on that virtual drive. The set consists of two active volumes (/dev/sdd and /dev/sdc) and one spare (/dev/sde). The spare will be automatically activated as soon as one of the current data volumes becomes unavailable for whatever reason. It is always a good idea to have at least one spare.

It is important to understand the use and selection of the RAID algorithm as well as the consistency policy.

7.3 Consistency policy

The consistency policy can determine how a raidset is rebuild in case of a failure. The policy is assigned during creation of the raidset with the ***"–consistency-policy="*** parameter. This is especially important in large (as in hundreds of gigabytes) raidsets. A policy of *resync* for example will totally re-synchronise the entire volume. You can imagine that if you have a 10 terabyte volume this may take a while. Not only is the rebuild process very long leaving you in a vulnerable position if an additional drive in that set fails. A journal approach may in this situation be of most benefit as the data only needs to be re-applied. A bitmap is a bit of an intermediate solution where only the stripes (parts of the data) need to be read from the "correct" disk and re-written to the disk(s) that are now in recovery mode.

- resync - Full resync is performed and all redundancy is regenerated when the array is started after unclean shutdown.
- bitmap - Resync assisted by a write-intent bitmap. Implicitly selected when using –bitmap.
- journal - For RAID levels 4/5/6, journal device is used to log transactions and replay after unclean shutdown. Implicitly selected when using –write-journal.
- ppl - For RAID5 only, Partial Parity Log is used to close the write hole and eliminate resync. PPL is stored in the metadata region of RAID member drives, no additional journal drive is needed.

Changing the consistency policy is only available in so called "Grow mode" and even that is somewhat restricted. Please refer to the up-to-date documentation at any given point in time as this may change.

7.4 Failures

" *So what is then being flagged as a failure?* " you'd ask. The first one is very obvious and that is basically a full hardware failure. A hardware failure can also be two-fold either

being the drive itself is defective or in the way the firmware on the disk-drive determines that there are so many bad-sectors on that drives it will flag itself as faulty. The first one renders the disk totally unusable and the second one allows a raid-algorithm to still use the disk for read/recovery purposes to verify and copy the data to a spare disk.
The second failure is any sort of write failure. If a write failure occurs on a device the `md` driver will immediately evict the drive and, if one is configured, access the spare to be included in the raid-set and start the recovery process. If no spare drive is configured or the spare drive is not a valid one for recovery purposes the raid set will be in a degraded state.
Another failure is inherent to RAID and that is that an any given point in time two or more pieces of data (whether real data or the parity) need to be physically written to disk. This is in almost all cases an a-synchronous process where there is a slight delay between when the first data is written and the second (or last) is completed. If at any stage a system failure (power, kernel panic, hardware) occurs the raidset is in an inconsistent state and is flagged as failed. When the system restarts and the MD driver checks the individual disks it determines the raidset is in a *dirty* state. Be aware that the moment a raidset is made available to the OS it always sits in a *dirty* state. Only a proper shutdown of the system will mark the raidset clean and no action is done when the system is started.

7.5 Data validation

If the raidset contains a large amount of data it is important to have the ability to validate if all data is still physically on disk as it was written in the past. Unlike filesystems, such as *BTRFS* and *ZFS*, there is no automatic validation or checksumming done. This may at some stage lead to a phenomenon called *bitrot*. Due to the changing characteristics of the physical substances on disk a bit may flip from 1 to 0 or vice versa. This is not new and with the advance in chemical engineering as well as recovery firmware in the drives themselves it is far less frequent than it did in the past but not impossible. The way a MD raidset can be checked (or repaired) is by writing "check" or "repair" to the ***"sync_ action"*** file in the md sysfs entry.

```
opensuse:~ # echo check > /sys/block/md127/md/sync_action
opensuse:~ # cat /sys/block/md127/md/sync_action
check
opensuse:~ # mdadm --detail /dev/md/md1
/dev/md/md1:
           Version : 1.2
     Creation Time : Tue Jun 30 17:45:22 2020
        Raid Level : raid1
        Array Size : 8379392 (7.99 GiB 8.58 GB)
     Used Dev Size : 8379392 (7.99 GiB 8.58 GB)
      Raid Devices : 2
     Total Devices : 3
```

```
       Persistence : Superblock is persistent

       Update Time : Tue Jun 30 17:47:29 2020
             State : clean, checking  <<<<<<<<<<<
    Active Devices : 2
   Working Devices : 3
    Failed Devices : 0
     Spare Devices : 1

Consistency Policy : resync

      Check Status : 2% complete  <<<<<<<<<<<<

              Name : opensuse:md1  (local to host opensuse)
              UUID : 8a1c4a76:97b69b4a:51bb6931:5c3d70c7
            Events : 17

    Number   Major   Minor   RaidDevice State
       0       8       32        0      active sync   /dev/sdc
       1       8       48        1      active sync   /dev/sdd

       2       8       64        -      spare   /dev/sde
```

The result may sometimes come back as a false positive mainly on RAID1 or RAID10 sets. This can also be attributed to the fact that the write to one disk has not been completed yet whilst the read from the check reads the stripe from the other. This is most often seen with swap spaces being located on those raidsets. With normal data partitions is is less likely to occur.

The `check` and `repair` options are often summarised as `scrubbing`. This basically means that in a RAID1 or RAID10 set the invalid data is overwritten by the valid data of any of the other disks on that set. On RAID5/6 new parity blocks will be written where applicable.

Validation of blocks being out of sync can be seen when looking at the `mismatch_cnt` file in the `/sys/block/<device>/md/` folder. Simulating an error by overwriting large sections of one of the physical devices shows that during a check or repair the count increases.

```
opensuse:/sys/block/md127/md # echo check > sync_action
opensuse:/sys/block/md127/md # cat mismatch_cnt
81568
```

7.6 Recovery

If the MD driver determines a dirty state the driver will start the recovery of the raidset based on the before-mentioned consistency policy. If a spare drive is configured it will be made an active member of the raidset and the existing data of the remaining members will be rebuild. The newer kernels have a MD driver that has an inbuilt load-algorithm and will dynamically adjust rebuild IO's in order to not impact active, application driven, IO's. That also means that the recovery time of the raid-set is proportionally depending on the host/application IO activity. It is therefore almost impossible to determine how long a rebuild is going to take. This is not necessarily limited to the Linux md driver but goes for all raid based systems. Yes even large scale raid-arrays.

That being said there are a few knobs and switches you can use to improve rebuild times or adjust parameters to even slow down the rebuild process in order to minimise application performance impact. As there is a very fine line between what is acceptable by the application to still be able to run properly and business requirements regarding redundancy of data.

7.6.1 Adjusting raid synchronisation

If out-of-sync issues are observed and the consistency policy is set to `resync` you may want to adjust the resync speed as this can significantly hamper your infrastructure if you have build raid-sets from externally provisioned devices. The way to do this is by setting the `sync_speed_max` value to a number of KIBI/s (that is kiloBITS per second). The values can be set globally in `/proc/sys/dev/raid/speed_limit_{min,max}` or per raidset in `/sys/block/<raiddev>/sync_speed_{min,max}`. Be aware that if you set this too low and the write workload to the device is higher than the `sync_speed_max` the resynchronisation process will not catch up during that period and an even greater performance impact will be observed.

- During synchronisation the speed will be shown.

```
opensuse:/sys/block/md127/md # echo repair > sync_action
opensuse:/sys/block/md127/md # cat sync_speed
12394
opensuse:/sys/block/md127/md # cat sync_min
0
opensuse:/sys/block/md127/md # cat sync_max
max
```

- The global settings for the `md` driver.

```
opensuse:~ # cat /proc/sys/dev/raid/speed_limit_max
200000
```

```
opensuse:~ # cat /proc/sys/dev/raid/speed_limit_min
1000
```

The above can be set permanently via `sysctl` with the appropriate values.

```
opensuse:~ # sysctl -a | grep dev.raid
dev.raid.speed_limit_max = 15000
dev.raid.speed_limit_min = 1000

opensuse:~ # sysctl -w dev.raid.speed_limit_max=20000
dev.raid.speed_limit_max = 20000
```

Be aware these are "targeted" values. It will do it's best to achieve them but the process is depending on existing workloads to the array.

7.7 Correcting failed raidsets

The fact of life is that hardware is subject to failures. Whether this being manufacturing issues or just simple wear & tear, sooner or later it will fail. Having a replacement available is therefore important. The `md` driver supports hot-adding devices to an existing raidset. It is recommended to have one or more spare drives available depending on the number and size of your raid-sets.

The below raid-set does not have a "hot-spare" available.

```
opensuse:~ # mdadm --detail /dev/md/raid5
/dev/md/raid5:
           Version : 1.2
     Creation Time : Tue Nov 10 15:35:01 2020
        Raid Level : raid5
        Array Size : 1619200 (1581.25 MiB 1658.06 MB)
     Used Dev Size : 809600 (790.63 MiB 829.03 MB)
      Raid Devices : 3
     Total Devices : 3
       Persistence : Superblock is persistent

       Update Time : Tue Nov 10 16:50:11 2020
             State : clean
    Active Devices : 3
   Working Devices : 3
    Failed Devices : 0
     Spare Devices : 0

            Layout : left-symmetric
```

```
        Chunk Size : 64K

Consistency Policy : resync

              Name : opensuse:raid5  (local to host opensuse)
              UUID : a33c76be:fd84996d:8756f7a8:bd7b602f
            Events : 56

    Number   Major   Minor   RaidDevice State
       0       8       16        0      active sync   /dev/sdb
       1       8       32        1      active sync   /dev/sdc
       2       8       48        2      active sync   /dev/sdd
```

To add one use the mdadm tool.

```
opensuse:~ # mdadm --manage /dev/md/raid5 --add-spare /dev/sde
mdadm: added /dev/sde
opensuse:~ # mdadm --detail /dev/md/raid5
/dev/md/raid5:
           Version : 1.2
     Creation Time : Tue Nov 10 15:35:01 2020
        Raid Level : raid5
        Array Size : 1619200 (1581.25 MiB 1658.06 MB)
     Used Dev Size : 809600 (790.63 MiB 829.03 MB)
      Raid Devices : 3
     Total Devices : 4
       Persistence : Superblock is persistent

       Update Time : Tue Nov 10 17:30:56 2020
             State : clean
    Active Devices : 3
   Working Devices : 4
    Failed Devices : 0
     Spare Devices : 1

            Layout : left-symmetric
        Chunk Size : 64K

Consistency Policy : resync

              Name : opensuse:raid5  (local to host opensuse)
              UUID : a33c76be:fd84996d:8756f7a8:bd7b602f
            Events : 57
```

```
    Number   Major   Minor   RaidDevice State
       0       8       16        0      active sync   /dev/sdb
       1       8       32        1      active sync   /dev/sdc
       2       8       48        2      active sync   /dev/sdd

       3       8       64        -      spare   /dev/sde
```

When one of the other devices observes a failure the hot-spare will immediately take its place and the resynchronisation is executed immediately.

When the drive `/dev/sdd` fails you'll see the hot-spare moving in immediately.

```
opensuse:/sys/block/md127/md/dev-sdd # mdadm --detail /dev/md/raid5
/dev/md/raid5:
           Version : 1.2
     Creation Time : Tue Nov 10 15:35:01 2020
        Raid Level : raid5
        Array Size : 1619200 (1581.25 MiB 1658.06 MB)
     Used Dev Size : 809600 (790.63 MiB 829.03 MB)
      Raid Devices : 3
     Total Devices : 4
       Persistence : Superblock is persistent

       Update Time : Tue Nov 10 17:55:08 2020
             State : clean, degraded, recovering
    Active Devices : 2
   Working Devices : 3
    Failed Devices : 1
     Spare Devices : 1

            Layout : left-symmetric
        Chunk Size : 64K

Consistency Policy : resync

    Rebuild Status : 13% complete

              Name : opensuse:raid5  (local to host opensuse)
              UUID : a33c76be:fd84996d:8756f7a8:bd7b602f
            Events : 61

    Number   Major   Minor   RaidDevice State
       0       8       16        0      active sync   /dev/sdb
```

```
       1       8       32        1      active sync   /dev/sdc
       3       8       64        2      spare rebuilding   /dev/sde

       2       8       48        -      faulty   /dev/sdd
```

When the raid-set does not have a hot-spare it will change into a degraded state. This not only results in lack of redundancy but will also incur a significant performance impact.

```
opensuse:~ # mdadm --detail /dev/md/raid5
/dev/md/raid5:
           Version : 1.2
     Creation Time : Tue Nov 10 17:56:52 2020
        Raid Level : raid5
        Array Size : 1619200 (1581.25 MiB 1658.06 MB)
     Used Dev Size : 809600 (790.63 MiB 829.03 MB)
      Raid Devices : 3
     Total Devices : 3
       Persistence : Superblock is persistent

       Update Time : Tue Nov 10 17:58:13 2020
             State : clean, degraded
    Active Devices : 2
   Working Devices : 2
    Failed Devices : 1
     Spare Devices : 0

            Layout : left-symmetric
        Chunk Size : 64K

Consistency Policy : resync

              Name : opensuse:raid5  (local to host opensuse)
              UUID : 5f1912f0:81a9a6dc:5f108b58:bfad5982
            Events : 61

    Number   Major   Minor   RaidDevice State
       0       8       16        0      active sync   /dev/sdb
       1       8       32        1      active sync   /dev/sdc
       -       0        0        2      removed

       2       8       48        -      faulty   /dev/sdd
```

There is no such thing as a `global spare` in the `md` context. This mean you cannot have one spare device for multiple raid-sets. A workaround may be to periodically poll the state of

the raid-set and depending on that add a device as spare.

In the above example when the `/dev/sde` drive gets added to the raid-set it immediately starts rebuilding.

```
opensuse:~ # mdadm --manage /dev/md/raid5 --add /dev/sde
mdadm: added /dev/sde
opensuse:~ # mdadm --detail /dev/md/raid5
/dev/md/raid5:
           Version : 1.2
     Creation Time : Tue Nov 10 17:56:52 2020
        Raid Level : raid5
        Array Size : 1619200 (1581.25 MiB 1658.06 MB)
     Used Dev Size : 809600 (790.63 MiB 829.03 MB)
      Raid Devices : 3
     Total Devices : 4
       Persistence : Superblock is persistent

       Update Time : Tue Nov 10 18:06:45 2020
             State : clean, degraded, recovering
    Active Devices : 2
   Working Devices : 3
    Failed Devices : 1
     Spare Devices : 1

            Layout : left-symmetric
        Chunk Size : 64K

Consistency Policy : resync

    Rebuild Status : 8% complete

              Name : opensuse:raid5  (local to host opensuse)
              UUID : 5f1912f0:81a9a6dc:5f108b58:bfad5982
            Events : 64

    Number   Major   Minor   RaidDevice State
       0       8       16        0      active sync   /dev/sdb
       1       8       32        1      active sync   /dev/sdc
       3       8       64        2      spare rebuilding   /dev/sde

       2       8       48        -      faulty   /dev/sdd
```

7.8 DMRAID

As an alternative to MD there is an alternative called DMRAID. As the name indicates this is part of the Device Mapper architecture and as such creates a device in the `/dev` directory with a `dm-x` where x is a sequence number.

To be honest this implementation is almost not used (at least I haven't seen it often).

Chapter 8

Block devices

A block device is basically the lowest level of a contiguous range of addressable space presented out of a physical device. Everything that has been mentioned in the previous chapters sits on top of these block devices to enable all sorts of functionality. A block device is not necessarily tied to a physical device as a single entity as it may also be a SCSI Logical Unit or NVMe ANA that is presented out of an array or other sort of controller. It may also be presented over a network like for example a iSCSI device.

The characteristics of individual devices may differ very much. The capabilities that can be provided by the device is communicated via so called inquiry commands. The responses that come back from the device enables the OS layer to use certain features. An example is clustering where multiple hosts may access the same device and certain locking mechanisms like reserves may need to be provided by the disk.

As you may know everything in Linux is represented as a *file*. Whether it is a overview of memory usage (Try `cat /proc/meminfo`) CPU (`cat /proc/cpuinfo`) or a peripheral like a diskdrive.

In earlier chapters you have seen me referring to addressable units like `/dev/sda` or `/dev/sdc`. The use of these addressable units in examples for this book is no problem. It basically is a virtual machine encapsulated in VirtualBox (or ESX or KVM, doesn't matter) which doesn't change so every time I reboot the entire, virtual, hardware stack will remain the same. It is important to understand that in an environment where hardware representation changes (like for instance in SAN's) these addressable units may change the name and thus may at some stage point to a different device.

It is therefore important that if you have management applications that depend on certain characteristics or functions of a device that these are not pointing to that local name but to the actual hardware path or even better the UUID that gets presented out of that hardware

device.

8.1 Device naming

For storage there are basically two identifiers that are most interesting from a device naming perspective. The identifiers that are tied to the physical device and the identifiers tied to a logical entity, such as a file-system, residing on that device.

Device identifiers are the WWID, partition GUUID and serial number. The file system identifiers are the UUID and the file-system label. Device identifiers do not change when they are re-created. A disk device presented out of a SAN over the same path will still retain the same device WWID. If you however re-write a file-system on that device a new UUID will be created and you would need to update your fstab to have it automatically mounted.

As an example a `/dev/sdc` disk is represented by a device presented via a system path and the type/model of that device determines how `udev` is handling the naming convention. If something changes in the device path or the order how it is discovered, that same physical device might be named `/dev/sdx`. There is no way to tell just based on the `/dev/sd` entry which physical device it actually is referring to.

If you have disks presented out of a san this will incur a significant problem when applications use the device based on the `/dev/sdx`name. For example lets say your backup application utilises a tape device via `/dev/sta`, which is located on the same site as the server, and `/dev/stb` locate on a remote site and after a reboot the discovery order changes resulting in the device naming to be flipped you may have two problems. Either you backup application is not able to correctly mount the tape or all you backup and restore operations will all of a sudden traverse the inter switch links (ISL's) between the two sites. When tape-libraries are involved it becomes even more cumbersome as the backup application is not able to instruct the library to move the correct tapes around to and from the drives. (Yes, I've been there and it resulted in the backup application vendor to withdraw their tool). Takeaway is to **ALWAYS** use UUID's or unique entities of a device that do not change everywhere where it is important.

The disk devices that are subject to rules by `udev` comes from the inventory in `sysfs`. These are then outlined by `udev` according to a few characteristics and `udev rules` and subsequently listed in `/dev/disk` under various characteristics.

As an example:

```
[root@opensuse ~]# ll /dev/disk/
total 0
drwxr-xr-x. 2 root root 2300 Dec 29 16:38 by-id
drwxr-xr-x. 2 root root   60 Dec 29 16:38 by-label
drwxr-xr-x. 2 root root   60 Dec 29 16:38 by-partlabel
drwxr-xr-x. 2 root root  140 Dec 29 16:38 by-partuuid
```

```
drwxr-xr-x. 2 root root 2580 Dec 29 16:45 by-path
drwxr-xr-x. 2 root root  280 Dec 29 16:38 by-uuid
```

The `sysfs` entries look like this

```
opensuse:~ # tree /sys/block/
/sys/block/
|- dm-0 -> ../devices/virtual/block/dm-0
|- dm-1 -> ../devices/virtual/block/dm-1
|- dm-2 -> ../devices/virtual/block/dm-2
|- dm-3 -> ../devices/virtual/block/dm-3
|- dm-4 -> ../devices/virtual/block/dm-4
|- nvme0n1 -> ../devices/pci0000:00/0000:00:0e.0/nvme/nvme0/nvme0n1
|- nvme0n2 -> ../devices/pci0000:00/0000:00:0e.0/nvme/nvme0/nvme0n2
|- nvme0n3 -> ../devices/pci0000:00/0000:00:0e.0/nvme/nvme0/nvme0n3
|- nvme0n4 -> ../devices/pci0000:00/0000:00:0e.0/nvme/nvme0/nvme0n4
|- sda -> ../devices/pci0000:00/0000:00:14.0/host2/target2:0:0/2:0:0:0/block/sda
|- sdb -> ../devices/pci0000:00/0000:00:14.0/host2/target2:0:1/2:0:1:0/block/sdb
|- sdc -> ../devices/pci0000:00/0000:00:14.0/host2/target2:0:2/2:0:2:0/block/sdc
|- sdd -> ../devices/pci0000:00/0000:00:14.0/host2/target2:0:3/2:0:3:0/block/sdd
|- sde -> ../devices/pci0000:00/0000:00:14.0/host2/target2:0:4/2:0:4:0/block/sde
|- sr0 -> ../devices/pci0000:00/0000:00:01.1/ata2/host1/target1:0:0/1:0:0:0/block/sr0
```

As you can see this shows the paths different devices take in the subsystem which is very handy for troubleshooting especially when larger storage subsystems are attached. Based on information out of the various device drivers `udev` is pulling this together and creates the usable device pointers in `/dev`.

For example a `udevadm info` output of the `nvme0n1` device shows:

```
opensuse:~ # udevadm info /dev/nvme0n1
P: /devices/pci0000:00/0000:00:0e.0/nvme/nvme0/nvme0n1
N: nvme0n1
L: 0
S: disk/by-id/lvm-pv-uuid-np7uD7-Krzd-YHD4-qG2D-xr0X-gYms-t3tXG1
S: disk/by-id/nvme-ORCL-VBOX-NVME-VER12_VB1234-56789
S: disk/by-path/pci-0000:00:0e.0-nvme-1
S: disk/by-id/nvme-eui.5abb44353d4b7a4a9b248357b2fa67f5
E: DEVPATH=/devices/pci0000:00/0000:00:0e.0/nvme/nvme0/nvme0n1
E: DEVNAME=/dev/nvme0n1
E: DEVTYPE=disk
E: MAJOR=259
E: MINOR=0
E: SUBSYSTEM=block
E: USEC_INITIALIZED=15309236
```

```
E: ID_WWN=eui.5abb44353d4b7a4a9b248357b2fa67f5
E: MPATH_SBIN_PATH=/sbin
E: DM_MULTIPATH_DEVICE_PATH=0
E: ID_SERIAL_SHORT=VB1234-56789
E: ID_MODEL=ORCL-VBOX-NVME-VER12
E: ID_REVISION=1.0
E: ID_SERIAL=ORCL-VBOX-NVME-VER12_VB1234-56789
E: ID_PATH=pci-0000:00:0e.0-nvme-1
E: ID_PATH_TAG=pci-0000_00_0e_0-nvme-1
E: ID_FS_UUID=np7uD7-Krzd-YHD4-qG2D-xr0X-gYms-t3tXG1
E: ID_FS_UUID_ENC=np7uD7-Krzd-YHD4-qG2D-xr0X-gYms-t3tXG1
E: ID_FS_VERSION=LVM2 001
E: ID_FS_TYPE=LVM2_member
E: ID_FS_USAGE=raid
E: COMPAT_SYMLINK_GENERATION=2
E: SYSTEMD_READY=1
E: SYSTEMD_ALIAS=/dev/block/259:0
E: SYSTEMD_WANTS=lvm2-pvscan@259:0.service
E: DEVLINKS=/dev/disk/by-id/lvm-pv-uuid-np7uD7-Krzd-YHD4-qG2D-xr0X-gYms-t3tXG1 \
  /dev/disk/by-id/nvme-ORCL-VBOX-NVME-VER12_VB1234-56789 \
  /dev/disk/by-path/pci-0000:00:0e.0-nvme-1 \
  /dev/disk/by-id/nvme-eui.5abb44353d4b7a4a9b248357b2fa67f5
E: TAGS=:systemd:
```

From a troubleshooting perspective there is not very much you can do on this level. The main issues are most often with discovery and presentation. These problems should be investigated in `udev`, the presentation layer like for example fibre-channel or iSCSI or even the lower level firmware and drivers for the hardware attached to the devices or infrastructure.

When it comes to identifying issues the tools in the various Linux packages can provide a treasure trove of information. In order to use these verify if the `sg3-utils` and `nvmecli` packages are installed. Most, if not all distributions have these in their repositories.

8.2 Identifying device characteristics

When a storage controller presents a disk to the system various triggers kick off to identify the disk and make an inventory of its characteristics. Depending on the device type it may use SCSI, nvme or other inquiry commands.

8.2.1 SCSI

Lets start with a SCSI device. (We'll come back to the protocol later on.) The command is actually named `inquiry` and what it does initially is obtaining a list of LUNS (i.e. the

addressable devices) supported, so called, VPD's which stands for `Vital Product Data`. The VPD's are presented in pages where each page has an ID and a predefined list of which information it should provide. All devices that claim to adhere to the SCSI standard should respond with at least two pages `0x00` and `0x83`. The first one is the device identification page and the `0x83` page is a list of all other VPD pages the device supports. Subsequent inquiry commands with these VPD page requests can then be sent to the device to obtain more information.

The `sg-utils` [23] package contain a substantial library of tools that use the SCSI command set to perform a myriad of functions on a large as variety of equipment.

Be aware that all these tools operate on a very low level and can/will bypass restrictive measures that are put in place by any higher level mechanism. In other words it can/will destroy your data without asking for permission or confirmation.

The `sg_inq` tool provides the following on a standard disk that is presented out of Virtual Box

```
opensuse:~ # sg_inq /dev/sdc
standard INQUIRY:
  PQual=0  Device_type=0  RMB=0  LU_CONG=0  version=0x05  [SPC-3]
  [AERC=0]  [TrmTsk=0]  NormACA=0  HiSUP=0  Resp_data_format=0
  SCCS=0  ACC=0  TPGS=0  3PC=0  Protect=0  [BQue=0]
  EncServ=0  MultiP=0  [MChngr=0]  [ACKREQQ=0]  Addr16=0
  [RelAdr=0]  WBus16=1  Sync=0  [Linked=0]  [TranDis=0]  CmdQue=1
    length=36 (0x24)   Peripheral device type: disk
 Vendor identification: VBOX
 Product identification: HARDDISK
 Product revision level: 1.0
```

The VPD pages that are supported on a virtual disk are negligible as the underlying hardware is not directly exposed and only very basic functionality is therefore provided. All protocol related support is obfuscated by the virtualisation stack and is handled on that level.

```
opensuse:~ # sg_vpd /dev/sdc
Supported VPD pages VPD page:
 Device identification [di]
```

When executing this on physical devices it becomes much more interesting.

```
[ server % ~ ] sudo sg_inq /dev/sdc
standard INQUIRY:
  PQual=0  Device_type=0  RMB=0  LU_CONG=0  version=0x05  [SPC-3]
  [AERC=0]  [TrmTsk=0]  NormACA=0  HiSUP=0  Resp_data_format=2
  SCCS=0  ACC=0  TPGS=0  3PC=0  Protect=0  [BQue=0]
```

```
  EncServ=0  MultiP=0  [MChngr=0]  [ACKREQQ=0]  Addr16=0
  [RelAdr=0]  WBus16=0  Sync=0  [Linked=0]  [TranDis=0]  CmdQue=1
  [SPI: Clocking=0x0  QAS=0  IUS=0]
    length=96 (0x60)   Peripheral device type: disk
 Vendor identification: ATA
 Product identification: ST2000DL003-9VT1
 Product revision level: CC32
 Unit serial number:             5YD0SEL5
```

The supported VPD pages:

```
[ server % ~ ] sudo sg_vpd /dev/sdc
Supported VPD pages VPD page:
  Supported VPD pages [sv]
  Unit serial number [sn]
  Device identification [di]
  ATA information (SAT) [ai]
  Block limits (SBC) [bl]
  Block device characteristics (SBC) [bdc]
  Logical block provisioning (SBC) [lbpv]
```

Each of these can subsequently be queried individually.

```
[ server % ~ ] sudo sg_vpd -p sn /dev/sdc
Unit serial number VPD page:
  Unit serial number:             5YD0SEL5

[ server % ~ ] sudo sg_vpd -p di /dev/sdc
Device Identification VPD page:
  Addressed logical unit:
    designator type: vendor specific [0x0],  code set: ASCII
      vendor specific:             5YD0SEL5
    designator type: T10 vendor identification,  code set: ASCII
      vendor id: ATA
      vendor specific: ST2000DL003-9VT166                                    5YD0SEL5
    designator type: NAA,  code set: Binary
      0x5000c5002f0289a9

[ server % ~ ] sudo sg_vpd -p ai /dev/sdc
ATA information VPD page:
  SAT Vendor identification: linux
  SAT Product identification: libata
  SAT Product revision level: 3.00
  Device signature indicates SATA transport
```

```
  Command code: 0xec
  ATA command IDENTIFY DEVICE response summary:
    model: ST2000DL003-9VT166
    serial number:             5YD0SEL5
    firmware revision: CC32

[ server % ~ ] sudo sg_vpd -p bl /dev/sdc
Block limits VPD page (SBC):
  Write same non-zero (WSNZ): 0
  Maximum compare and write length: 0 blocks [Command not implemented]
  Optimal transfer length granularity: 1 blocks
  Maximum transfer length: 0 blocks [not reported]
  Optimal transfer length: 0 blocks [not reported]
  Maximum prefetch transfer length: 0 blocks [ignored]
  Maximum unmap LBA count: 0 [Unmap command not implemented]
  Maximum unmap block descriptor count: 0 [Unmap command not implemented]
  Optimal unmap granularity: 0 blocks [not reported]
  Unmap granularity alignment valid: false
  Unmap granularity alignment: 0 [invalid]
  Maximum write same length: 0 blocks [not reported]
  Maximum atomic transfer length: 0 blocks [not reported]
  Atomic alignment: 0 [unaligned atomic writes permitted]
  Atomic transfer length granularity: 0 [no granularity requirement
  Maximum atomic transfer length with atomic boundary: 0 blocks [not reported]
  Maximum atomic boundary size: 0 blocks [can only write atomic 1 block]

[ server % ~ ] sudo sg_vpd -p bdc /dev/sdc
Block device characteristics VPD page (SBC):
  Nominal rotation rate: 5900 rpm
  Product type: Not specified
  WABEREQ=0
  WACEREQ=0
  Nominal form factor not reported
  ZONED=0
  RBWZ=0
  BOCS=0
  FUAB=0
  VBULS=0
  DEPOPULATION_TIME=0 (seconds)

[ server % ~ ] sudo sg_vpd -p lbpv /dev/sdc
Logical block provisioning VPD page (SBC):
```

```
  Unmap command supported (LBPU): 0
  Write same (16) with unmap bit supported (LBPWS): 1
  Write same (10) with unmap bit supported (LBPWS10): 0
  Logical block provisioning read zeros (LBPRZ): 0
  Anchored LBAs supported (ANC_SUP): 0
  Threshold exponent: 0 [threshold sets not supported]
  Descriptor present (DP): 0
  Minimum percentage: 0 [not reported]
  Provisioning type: 0 (not known or fully provisioned)
  Threshold percentage: 0 [percentages not supported]
[ server % ~ ]
```

If you looked closely you will have seen a wwn in the device identification page. In this case for device `/dev/sdc/` it is `0x5000c5002f0289a9`. That number is than used to uniquely identify the disk and is then used to build the stack. When looking at the `/dev/disk/by-id` directory you will see the same wwn returning

```
[ server % ~ ] ll /dev/disk/by-id
<snip>
lrwxrwxrwx 1 root root    9 Apr 15 17:14 scsi-35000c5002f0289a9 -> ../../sdc <<<<<<<<<
<snip>
```

There are however more identifiers under which this disk is now known to the system and various tools and applications can use this accordingly.

```
[ server % ~ ] ll /dev/disk/by-id | grep sdc
lrwxrwxrwx 1 root root    9 Apr 15 17:25 ata-ST2000DL003-9VT166_5YD0SEL5 -> ../../sdc
lrwxrwxrwx 1 root root    9 Apr 15 17:25 scsi-0ATA_ST2000DL003-9VT1_5YD0SEL5 -> ../../sdc
lrwxrwxrwx 1 root root    9 Apr 15 17:25 scsi-1ATA_ST2000DL003-9VT166_5YD0SEL5 -> ../../sdc
lrwxrwxrwx 1 root root    9 Apr 15 17:25 scsi-35000c5002f0289a9 -> ../../sdc
lrwxrwxrwx 1 root root    9 Apr 15 17:25 scsi-SATA_ST2000DL003-9VT1_5YD0SEL5 -> ../../sdc
lrwxrwxrwx 1 root root    9 Apr 15 17:25 wwn-0x5000c5002f0289a9 -> ../../sdc>
```

8.2.1.1 Device capabilities

Let's have a look at some device capabilities that are presented out of the inquiry response.

As we saw earlier the response retrieved via the `sg_inq` tool is interpreted and then displayed:

```
standard INQUIRY:
 PQual=0  Device_type=0  RMB=0  LU_CONG=0  version=0x05  [SPC-3]`

  [AERC=0]  [TrmTsk=0]  NormACA=0  HiSUP=0  Resp_data_format=2
  SCCS=0  ACC=0  TPGS=0  3PC=0  Protect=0  [BQue=0]
  EncServ=0  MultiP=0  [MChngr=0]  [ACKREQQ=0]  Addr16=0
  [RelAdr=0]  WBus16=0  Sync=0  [Linked=0]  [TranDis=0]  CmdQue=1
```

```
  [SPI: Clocking=0x0  QAS=0  IUS=0]
    length=96 (0x60)   Peripheral device type: disk
 Vendor identification: ATA
 Product identification: ST2000DL003-9VT1
 Product revision level: CC32
 Unit serial number:             5YD0SEL5
```

When looking somewhat deeper and get the underlying hex values we can see the same information but are now able to associate this with the values as outlined by the T10 SCSI standard. (More on that later)

```
[ server % ~ ] sudo sg_inq -H /dev/sdc
00     00 00 05 02 5b 00 00 02  41 54 41 20 20 20 20 20    ....[...ATA
10     53 54 32 30 30 30 44 4c  30 30 33 2d 39 56 54 31    ST2000DL003-9VT1
20     43 43 33 32 00 00 00 00  00 00 00 00 00 00 00 00    CC32............
30     00 00 00 00 00 00 00 00  00 00 00 60 03 20 03 00    ...........`. ..
40     00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
50     00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ................
```

As you now have the absolute values you can associate these with the defined fields out of the T10 SCSI SPC documentation.

Table 81 — Standard INQUIRY data format

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	RMB	Reserved						
2	VERSION							
3	Obsolete	Obsolete	NORMACA	HISUP	RESPONSE DATA FORMAT			
4	ADDITIONAL LENGTH (n-4)							
5	SCCS	ACC	TPGS		3PC	Reserved		PROTECT
6	BQUE	ENCSERV	VS	MULTIP	MCHNGR	Obsolete	Obsolete	ADDR16[a]
7	Obsolete	Obsolete	WBUS16[a]	SYNC[a]	LINKED	Obsolete	CMDQUE	VS
8	(MSB)	T10 VENDOR IDENTIFICATION						
15								(LSB)
16	(MSB)	PRODUCT IDENTIFICATION						
31								(LSB)
32	(MSB)	PRODUCT REVISION LEVEL						
35								(LSB)
36	Vendor specific							
55								
56	Reserved				CLOCKING[a]		QAS[a]	IUS[a]
57	Reserved							
58	(MSB)	VERSION DESCRIPTOR 1						
59								(LSB)
	⋮							
72	(MSB)	VERSION DESCRIPTOR 8						
73								(LSB)
74	Reserved							
95								

Figure 8.1: SCSI SPC-3 Inquiry response

The standard requires at least a response of the first 36 bytes. Byte 4 will indicate how many additional bytes are appended to it from byte 4 onwards.

Be aware that standards are under constant development and fields may become obsolete or may be replaced with other meanings. It is therefore important that not only the version field is taken into account but also the driver and firmware versions on the respective hosts are up to date in order to be able to correctly use that device.

A few fields I do want you to take notice off.

- Byte 0
  - The first three bits are most often all 0's.
  - The device type in bits 0 to 4 depict what kind of device it is. Disk, Tape drive, tape library, printer etc... and also determine which peripheral command set should be used.
- Byte 1
  - The RMB bit stands for removable medium bit which, in basically all enterprise storage devices, is set to 0. In most cases when this is set to 1 all caching algorithms to and from this device are going to be disabled.
- Byte 2
  - This is one of the most important ones as this one basically enforced which version of the peripheral command set can be used from the host to the device and vice versa. This is very important as many functions like for example locking and reservation commands are very different between versions and many newer features depend on it. At the time of this writing it should always be set to 0x05 or higher (for disk devices) as that depicts the SPC-3, 4 and 5 command set support respectively.
- Byte 4
  - In our example this value was 0x5b (91 bytes).
- Bytes 5 6 and 7 are the ones that basically the on and off switches for the various device specific features.
  - SCCS - this bit indicates if an embedded storage controller component is available and supported. This allows applications to directly execute commands against the controller to do all sorts of functions like provisioning, replication etc.
  - TPGS - Target Port Group Support is used for multipathing access with different access characteristics. You may have come across the term ALUA (A-synchronous Logical Unit Access) which is using these two bits. We'll dedicate a separate chapter on multipathing.
  - 3PC - The 3PC bit informs the block layer that 3rd Party Copy commands are supported. This allows applications to send commands to the storage copy manager and offload data movement activities to that controller.
  - The Protect bit is used for the T10 data integrity function which we also touch on later.
  - The BQue and CmdQue bits are mutually exclusive and indicate if a Full or Basic taskmanagement commandset can be used.
  - The EncServ bit is sometimes used in storage arrays and tape libraries to indicate support for direct environmental diagnostics capabilities.

Depending on the issues you observe in your storage environment you may be able to tie certain characteristics to the information you obtained via one of the above methods. If for example you run into a problem with certain clustering solutions you may want to check if the correct SCSI version is presented out of a storage array. Very often the functions and features that are required are determined by certain configuration settings in the arrays or

controllers.

8.3 Caching

Depending on the implementation the way and level of caching can be done on many levels. The main decision point will always be where to use caching and to what extend caching is beneficial to the application response time. If caching is set too aggressive on, lets say the scheduler level, it may be negatively impacting caching algorithms on storage arrays.

As an example lets take a relative write intensive sequential workload. Storage array read-ahead caching algorithms look at the ratio where and how these write requests come in an may start pre-staging subsequent data in their caches. If one of the layers on the host system is also trying to do the same thing and balance out read and write requests, or write requests to different sectors on disk, this may interfere with the handling of cached data on the array. This may caused the cache-hitrate on the array to decline resulting in transactional delays to the application.

In order to see how well a host caching vs an array caching algorithm works you would need to obtain the `iostat` output and observe the `await`, `r_wait` and/or `w_wait` values as that would tell you the response time including the queuing delay. Compare these numbers against the cache hit and miss rate on the array. Experiment with different host settings against a workload profile that is representative for the application. Be aware that on an external storage array there are really dozens of knobs that can be flipped or tuned and that these systems most often have to serve a fairly diverse set of hosts. What might look excellent one days may show up a dreadful the other due to a change in access characteristics from other hosts or applications.

As I said in the beginning the tuning and optimisation of an IO subsystem from host to disk is somewhat beyond the scope of this book.

Chapter 9

Schedulers

As mentioned recent kernels provide only multi-queue schedulers. This is especially important since the rise of NVMe and NVMEoF (NVMe over Fabrics). Hardware has developed significantly over the last decade where spinning platers of magnetic material with an actuator head hovering over it is replaced by non-volatile silicon operating at lightning speeds and thus are not restricted by mechanical delays.

In order to facilitate this the older schedulers who where primarily optimized for single queue operations are now replaced so that a very large number of IO commands can be accepted by the software queues and be optimized to be send as fast as possible to the various hardware dispatch queues. The internal heuristics of each of these can be referred to below.

The main difference between the non-mq and mq schedulers is two-fold. First the mq schedulers have a far wider range of dispatch options and they are not as strictly bound to locality of the data on disk. The non-mq schedulers tried franticly to prevent movement of the disk heads by gathering IO requests and rearrange them in certain order so that these requests could be serviced by the disk or controller in the most optimal way. As that is no longer required due to the direct access to an individual area without any mechanical delay, the mq schedulers do not need a large list of so called `tunables` to be able to optimize IO's. That is not to say they cannot optimize IO request. Especially when it comes to different workload characteristics a scheduler like `BFQ` can be fairly beneficial. It needs to be said though that all schedulers do incur a latency penalty although in most cases this is negligible.

9.1 Non-MQ

deadline, noop, cfq are (or were) the available schedulers in older kernels. As mentioned these are no longer supported and should not be used.

9.2 MQ

The current multi-queue schedulers are predominantly the BFQ [24], Kyber [25] and the MQ-Deadline[26] scheduler.

As for troubleshooting the schedulers this is most often based on a trial and error basis as well as adjusting scheduler parameters, i.e. `tunables`, if required. Some schedulers use cgroups functionality to be able to adjust IO behaviour. Some distros use systemd in conjunction with cgroups to modify the schedulers. Refer to the documentation of you distribution how this is achieved.

An example on how to dynamically change the scheduler is shown below. At first you need to know which one is actually compiled into the kernel. A simple `cat` on the `scheduler` file of each device will show what is possible and which one is currently activated.

The one on the `sdc` drive is [bfq] at the moment.

```
[ server % ~ ] cat /sys/class/block/sdc/queue/scheduler
mq-deadline kyber [bfq] none
```

On the `nvme` drive there is no scheduler active/

```
[ server % ~ ] cat /sys/class/block/nvme0n1/queue/scheduler
[none] mq-deadline kyber bfq
```

To dynamically change this a simple `echo` command will do the trick:

```
[root@server ~]# echo kyber > /sys/class/block/nvme0n1/queue/scheduler

[root@server ~]# cat /sys/class/block/nvme0n1/queue/scheduler
mq-deadline [kyber] bfq none
```

In order to quickly and automatically select the scheduler of your choice a simple way is to use `udev` and adjust these values

As an example add the below in a file called `60-diskscheduler.conf` and save this in the `/etc/udev/rules.d` directory. *(Be aware the rules should be on one line. The \ is there otherwise it would not fit on the page)*

```
# set kyber scheduler for non-rotating disks
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0", \
  ATTR{queue/scheduler}="kyber"

# set bfq scheduler for rotating disks
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="1", \
  ATTR{queue/scheduler}="bfq"
```

Basically what udev does is looking for the `sd` devices and if the attribute `rotational` is set to 1 it modifies the scheduler to `bfq` and if it is set to 0 it will adjust the scheduler to `kyber`.

```
[root@server ~]# udevadm info -a /dev/sdc

  looking at device '/devices/pci0000:00/0000:00:17.0/ata4/host3/ \
      target3:0:0/3:0:0:0/block/sdc':
    KERNEL=="sdc"
    SUBSYSTEM=="block"
    DRIVER==""
    ATTR{alignment_offset}=="0"
    ATTR{capability}=="50"
<snip>
    ATTR{queue/read_ahead_kb}=="128"
    **ATTR{queue/rotational}=="1"** <<<<<<<<<<<<<<<<
    ATTR{queue/rq_affinity}=="1"
    **ATTR{queue/scheduler}=="mq-deadline kyber [bfq] none"**  <<<<<<<<<<<
    ATTR{queue/stable_writes}=="0"
    ATTR{queue/wbt_lat_usec}=="75000"
<snip>
```

9.3 Selection

So which scheduler should I use? In general the BFQ scheduler with the default options set are good for most general purpose applications. As with most processes in the Linux kernel there are a fair few knobs and switches with the `bfq` scheduler but you should really investigate and test your setup before implementing these changes into production systems. As the name (or acronym) implies the `bfq` scheduler acts as a `fair queuing` dispatcher. Each process requesting access to a devices is assigned a weight and a queue. That queue is budgeted to access the device at some stage depending on the configured tunables. By incrementing and decrementing the budget there will be a very well balanced IO profile achieved across all processes doing IO. The `bfq` scheduler tunables can be adjusted on a per device basis via the switches in the `/sys/block/<device>/queue/iosched/` directory or based on a `cgroup` configuration. For the latter you need to ensure this is enabled in the kernel via the `BFQ_GROUP_IOSCHED` flag.

```
Symbol: BFQ_GROUP_IOSCHED [=y]
 Type  : bool
 Defined at block/Kconfig.iosched:31
   Prompt: BFQ hierarchical scheduling support
   Depends on: BLOCK [=y] && IOSCHED_BFQ [=y] && BLK_CGROUP [=y]
   Location:
     -> IO Schedulers
 (2)   -> BFQ I/O scheduler (IOSCHED_BFQ [=y])
 Selects: BLK_CGROUP_RWSTAT [=y]
```

By default the `bfq` scheduler is more favourable to interactive and "real-time" applications. If your system is having issues on such a mixed workload you may want to investigate this. There is no way of telling what is right or wrong here upfront.

The `kyber` scheduler is simpler as it is a relatively straightforward one and it primarily tries to cater to get as close as possible to the `read_lat_nsec` and `write_lat_nsec` tunables irrespective of workload characteristics. This may be useful when a very pre-deterministic workload is offered from the application.

Obviously the choice is yours. Using individual devices can provide a more granular control over each disk whereas using cgroups gives a more manageable structure per process.

9.4 Tuning

The above is more related to generic systems administration and basically shows the quick and simple way to select and configure the appropriate scheduler. It will then need to be determined if and how additional tuning may be needed to optimize the respective scheduler via one or more `tunables` that may be available for the scheduler.

So when do you actually need to change these knobs? In general you don't need to at all. The defaults are pretty good for most workloads but some optimizations may be helpful in some cornercases.

The `kyber` scheduler has only two knobs to try an modify IO behaviour

```
[root@server iosched]# for i in *; do echo -n "$i  :"; cat $i;done
read_lat_nsec  :2000000
write_lat_nsec  :10000000
```

The `bfq` scheduler has a few more:

```
[root@server:/sys/block/sdd/queue/iosched]$ for i in *; do echo -n "$i  :"; cat $i;done
back_seek_max  :16384
back_seek_penalty  :2
fifo_expire_async  :250
fifo_expire_sync  :125
low_latency  :1
max_budget  :0
slice_idle  :8
slice_idle_us  :8000
strict_guarantees  :0
timeout_sync  :125
```

If your workload is primarly server related you may wish to disable the `low_latency` mode as that would prioritise interactive applications and ***realtime*** applications. If you want to fully control the bandwidth and distribution of the workload you would need to set this to

`0`. The `back_seek_max` and `back_seek_penalty` are primarily useful for rotational disk drives. It calculates, based on the diskdrive geometry, what is the optimum number of kilobytes to ***seek backwards*** in order to determine whether or not a drive head is close enough to the location of the current request for it to flag it as a next request. This behavior is also sometimes called ***read_ ahead requests***. It may need some testing based on your application and drive layout to find its optimum setting.

Be aware though that disks presented out of storage arrays may show up as rotational but these have their own algorithms to optimise workloads so fiddling with these parameters on those kind of disks is very likely to be futile.

The `fifo_expire_xxx` settings provide a way to prioritise synchronous v.s. a-synchronous workloads. This may be useful in both local and SAN-attached storage as many storage arrays have special algorithms that can detect this kind of workload differentiation and therefore also optimise its own internals to cater for this distinctive pattern.

Chapter 10

Protocols

In the storage-world there used to be a plethora of protocols which all regulated the communication between a storage device and a host. Only 2 of them have basically "survived" and one has been recently, in the sense of the last decade, introduced.

In the mainframe "big boys" league it is still a very much Ficon (ie SBCCS or Single Byte Command Code Set over Fibre Channel) and SCSI in the open systems world. NVMe is the latest addition that more or less takes care of the inherent limitations SCSI brings with it, and is very much optimized for dealing with very fast Solid State disk drives.

This is one of the larger and complex chapters as it also incorporates a lot of issues that are related to transport protocols like TCP/IP and Fibre Channel. Especially in environments where hosts obtain their disks from external systems, you often see issues which are in many cases very hard to diagnose. As it also has a lot of "horizontal" and "vertical" touch points you'll see that many sections show a various degree of issues which intertwine with each other. This makes is somewhat appear strange but when you look at it from a high level it all makes sense. It is imperative that when you attach a host to external storage you also have a relatively good understanding of what is involved from that side so you can work with your storage administrators in case issues arise.

10.1 Channels

As for the channel protocols on a Linux subsystem we more or less only have to look at SCSI and NVMe. The NVMe [27] part is a bit of a grey area as it crossed boundaries with memory, PCI, transport networks and interacts with the access characteristics of the individual device.

10.2 SCSI

SCSI could have a chapter on it's own. Let me rephrase that, it could have a book on its own when it comes to troubleshooting. The architecture is vast and is outlined in the, so called, SAM [1]. The acronym SCSI stands for `Small Computer Systems Interface` which already shows you the actual heritage it has. It was initially designed as a hardware and communications standard to be able to connect devices like printers, modems, keyboards etc. over a single bus to a PC. The fact that these days SCSI is still around and serving massive storage environments is really a testament to the original designers of the protocol who had the vision of creating a flexible and scalable method of device communications.

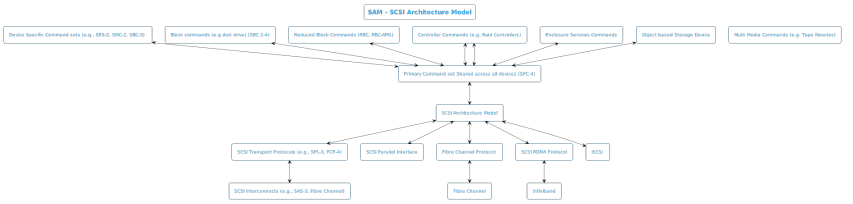


Figure 10.1: SCSI Architecture Model

The SAM model as shown has more protocols than depicted here however from a Linux troubleshooting perspective these are the most relevant ones.

When it comes to the protocol itself there is not really much you can do with regards to actually turn on knobs and move sliders. A few exceptions may be for example tuning queue-depth and supportability options. The protocol itself is pretty well fleshed out and is very stable.

The first thing is that the scsi driver itself is statically compiled in the kernel via the `sg` (SCSI generic) driver. It's parameters are :

- scatter_elem_sz:scatter gather element size (default: max(SG_SCATTER_SZ, PAGE_SIZE)) (int)
- def_reserved_size:size of buffer reserved for each fd (int)
- allow_dio:allow direct I/O (default: 0 (disallow)) (int)

I haven't come across major issues that would justify modifying any of these. In some corner cases where applications have control of the IO stack when it comes to caching, queueing etc. you may see that the `allow_dio` parameter is set to `1` but these are very rare.

In the "Block devices" chapter we've already seen the `lsblk` output and how the local devices can provide a plethora of information. The swiss-army knife on Linux when it comes to SCSI

[1] SCSI Architecture Model

is the `lsscsi` command utility which comes most often installed by default on both Redhat and SUSE. If not a `sudo dnf install lsscsi` or via one of your favourite package managers would to the trick.

So how does SCSI operate in general? SCSI works with so called Initiators and Targets. These are connected to a `scsi bus` and the initiator has the control over that bus. There can only be one active initiator on a bus. Anything that gets send to a device is done via a *write* command and anything that needs to be retrieved from a device is done via a *read* command from the initiator to the target. The targets themselves have no control over the bus. (There are a few exceptions but these cannot be controlled from a generic SCSI layer and depends on the device type and capabilities)

The read and write commands are not only used for storing and retrieving data but also for low level device communication. There are for example write commands that send instructions to a tape library to load a particular tape in a particular tape drive. Read commands are also used to obtain information from a device like for instance a serial number or any other characteristics. Depending on the amount of information required in the instruction set the read and write commands can be of different sizes. These are defined as READ(10), READ(12), READ(16), READ(32) and the same for write commands.

Whatever information is required from a device in order for it to be able to function will be determined via `inquiry` commands. The first inquiry command is more related to making an inventory of attached devices. This then allows the SCSI generic driver to build a "device tree" after which more targeted inventory commands provide a supported options list what the device actually is and what capabilities it can provide to the operating system. From there on the respective device drivers are loaded (if not already statically compiled in the kernel) and additional parameters can be set and used in order to actually use that specific device. The specifics of each device is obviously tied to the hardware capabilities but also what the driver allows you to do. If you have a shiny new raid-controller which has been on the shelf in the store for over a year you can rest assure that the driver comes shipped with the card in the box is already outdated and new or updated functions/features may already be available in newer releases.

The `inquiry` commands do request so called VPD's [2] of which one is mandatory and every SCSI device should return the data and that is a list of which other VPD's are supported. Based on that one or more subsequent requests will be made with additional VPD pages.

```
[root@server:~]$ sg_vpd -p sv /dev/sde
Supported VPD pages VPD page:
  Supported VPD pages [sv]
  Unit serial number [sn]
  Device identification [di]
  ATA information (SAT) [ai]
```

[2] Vital Product Data

```
  Block limits (SBC) [bl]
  Block device characteristics (SBC) [bdc]
  Logical block provisioning (SBC) [lbpv]
```

From there on the supported pages can be queried.

```
[root@server:~]$ sg_vpd -p di /dev/sde
Device Identification VPD page:
  Addressed logical unit:
    designator type: vendor specific [0x0],  code set: ASCII
      vendor specific:             Z4Z87FRT
    designator type: T10 vendor identification,  code set: ASCII
      vendor id: ATA
      vendor specific: ST2000DM006-2DM164                                  Z4Z87FRT
    designator type: NAA,  code set: Binary
      0x5000c500a30b35d1
```

The info that is retrieved via these inquiry commands on VPD pages are defined by the T10 standards body. There are however provisions that allow vendors to add functionality based on their own requirements. The information that is retrieved there is often in some binary or hex blob which does not make much sense if you don't know how to decode this. Many array vendors use these fields for functions and features that are proprietary to their solutions like, for example, remote replication or interaction with management agents that can be installed on hosts.

Next to the `READ`, `WRITE` and `Inquiry` commands there is a hole plethora of commands that handle the entire operation of the initiator and target management facilities. This can extend to upgrading firmware, doing inline copies of data between luns without host involvement, formatting of drives, getting status information on various levels etc. etc.

The command flow is explained a little later in this chapter in the Fibre Channel section.

So how does this help me? you can ask. The `inquiry` information that is being returned from the devices helps in identifying if you have the correct device in case you need to troubleshoot any issues. As I mentioned in the `block devices` chapter, you cannot rely on the logical device name that Linux gives to the device as that may change in various ways. It would be very detrimental to your system if you start troubleshooting the `/dev/sdgg/` disk which you think comes out of raid-array A where the actual disk is represented out of raid-array B. That is just an example and most of the time you're pretty sure where disks come from however when architectures become more complex and large SAN's are used in combination with a variety of Linux options, you're better be safe than sorry.

When it comes to terminology you will often see the acronym `HBTL` or `HCTL` which stands for `Host`, `BUS`/`Channel`, `Target`, `LUN`. The host in this case is not the server but the controller that is interacting with the operating system. The `BUS` is more or less synonymous to a channel. On this channel you can have one or more `Targets` such as array ports, tape libraries and

printers. Each target can present one or more LUNS [3]. A LUN can be any sort of device, physical disks, logical disks presented out of arrays, tape library robotics and tape drives, you name it.

The majority of error logging facilities will very often show this addressing schema. The below snippet shows an example.

```
/var/log/messages-20220619.gz:Jun 12 23:11:09 monster kernel: [48480.070957] sd 3:0:0:0: \
  [sdc] tag#23 FAILED Result: hostbyte=DID_BAD_TARGET driverbyte=DRIVER_OK cmd_age=117s
/var/log/messages-20220619.gz:Jun 12 23:11:09 monster kernel: [48480.070959] sd 3:0:0:0: \
  [sdc] tag#23 CDB: Write(10) 2a 00 72 54 18 10 00 00 08 00
```

The part after `sd` showing `3:0:0:0` is the representation of that addressing schema. If you know this you can easily trace that back to the actual device.

```
$ lsscsi -L
<snip>
[3:0:0:0]    disk    ATA      ST2000DL003-9VT1 CC32  /dev/sdc
  device_blocked=0
  iocounterbits=32
  iodone_cnt=0x2897
  ioerr_cnt=0x476
  iorequest_cnt=0x28f6
  queue_depth=32
  queue_type=simple
  scsi_level=6
  state=running
  timeout=30
<snip>
```

Keen readers would have noticed the above is not actually a SCSI drive but an ATA one. Linux uses the `libata` [28] library to ***tunnel*** or ***translate*** ATA commands over a SCSI command structure. This way the command set of SCSI's SAT [4] standard can be used to address ATA disk drives.

I would urge you to explore the `lsscsi` command in-depth as it provides very useful information for quick analysis of device characteristics and device paths which improves accuracy and speed in diagnosing issues.

10.2.1 Logging

The Linux kernel SCSI subsystem does provide an option to increase the verbosity of the logging facility.

[3] Logical Unit Numbers
[4] SCSI ATA Translation


Be aware that increasing the logging verbosity may incur a significant performance penalty and may even cause the system to come to a grinding halt. Do this only in circumstance when you suspect a problem within the IO stack and other means have been exhausted.

The logging verbosity may be set dynamically by echoing a value in `/proc/sys/scsi/logging_level` or statically via the `sysctl -a dev.scsi_logging_level=xxxxxx`command. Be aware that the latter will also be applied when starting up the system so you may see an additional delay there.

The facility provides a few trace-points, for a lack of a better word.

The source code shows a good explanation on how this is build up:

```
/*
 * This defines the scsi logging feature.  It is a means by which the user can
 * select how much information they get about various goings on, and it can be
 * really useful for fault tracing.  The logging word is divided into 10 3-bit
 * bitfields, each of which describes a loglevel.  The division of things is
 * somewhat arbitrary, and the division of the word could be changed if it
 * were really needed for any reason.  The numbers below are the only place
 * where these are specified.  For a first go-around, 3 bits is more than
 * enough, since this gives 8 levels of logging (really 7, since 0 is always
 * off).  Cutting to 2 bits might be wise at some point.
 */

#define SCSI_LOG_ERROR_SHIFT             0
#define SCSI_LOG_TIMEOUT_SHIFT           3
#define SCSI_LOG_SCAN_SHIFT              6
#define SCSI_LOG_MLQUEUE_SHIFT           9
#define SCSI_LOG_MLCOMPLETE_SHIFT        12
#define SCSI_LOG_LLQUEUE_SHIFT           15
#define SCSI_LOG_LLCOMPLETE_SHIFT        18
#define SCSI_LOG_HLQUEUE_SHIFT           21
#define SCSI_LOG_HLCOMPLETE_SHIFT        24
#define SCSI_LOG_IOCTL_SHIFT             27

#define SCSI_LOG_ERROR_BITS              3
#define SCSI_LOG_TIMEOUT_BITS            3
#define SCSI_LOG_SCAN_BITS               3
#define SCSI_LOG_MLQUEUE_BITS            3
#define SCSI_LOG_MLCOMPLETE_BITS         3
#define SCSI_LOG_LLQUEUE_BITS            3
```

```
#define SCSI_LOG_LLCOMPLETE_BITS          3
#define SCSI_LOG_HLQUEUE_BITS             3
#define SCSI_LOG_HLCOMPLETE_BITS          3
#define SCSI_LOG_IOCTL_BITS               3
```

So each of the keyword above can be set to a value from 0 to 7. If for example you need to increase verbosity on the ERROR, TIMEOUT and IOCTL bits with respective values of 2, 4 and 5 you would set the value to `500000042`. The command then to enter is `echo 402653201 > /proc/sys/dev/scsi/logging_level`

Shouldn't this be `5000000042`???)...

The value is specified as an octal number so 50000000042 to octal is 402653201. It's a bit cumbersome to do this manually so the good people of IBM have written a script called `scsi_logging_level` and is part of the `sg3_utils` package. Install that and you'll be able to use a simpler way of just specifying the values you want.

```
sserver:~ # scsi_logging_level -h
Usage: scsi_logging_level [OPTIONS]

Create, get or set scsi logging level.

Options:

        -h, --help       print this help
        -v, --version    print version information
        -s, --set        create and set logging level as specified on
                         command line
        -g, --get        get current logging level and display it
        -c, --create     create logging level as specified on command line
        -a, --all        specify value for all SCSI_LOG fields
        -E, --error      specify SCSI_LOG_ERROR
        -T, --timeout    specify SCSI_LOG_TIMEOUT
        -S, --scan       specify SCSI_LOG_SCAN
        -M, --midlevel   specify SCSI_LOG_MLQUEUE and SCSI_LOG_MLCOMPLETE
            --mlqueue    specify SCSI_LOG_MLQUEUE
            --mlcomplete specify SCSI_LOG_MLCOMPLETE
        -L, --lowlevel   specify SCSI_LOG_LLQUEUE and SCSI_LOG_LLCOMPLETE
            --llqueue    specify SCSI_LOG_LLQUEUE
            --llcomplete specify SCSI_LOG_LLCOMPLETE
        -H, --highlevel  specify SCSI_LOG_HLQUEUE and SCSI_LOG_HLCOMPLETE
            --hlqueue    specify SCSI_LOG_HLQUEUE
            --hlcomplete specify SCSI_LOG_HLCOMPLETE
        -I, --ioctl      specify SCSI_LOG_IOCTL
```

```
Exactly one of the options "-c", "-g" and "-s" has to be specified.
Valid values for SCSI_LOG fields are integers from 0 to 7.
```

What you actually need to enter here when it comes to which values on which parameters will be mentioned by your support vendor. When it comes to diagnosing IO errors you'll most often start of with an increase of logging on the `SCSI_LOG_ERROR` and `SCSI_LOG_TIMEOUT`. This will tell your support vendor where to dive in deeper. If the suspicion is to be on the device layer from a protocol perspective the `SCSI_LOG_IOCTL` will likely provide more insight in the command and response sequences. When it comes to diagnosing timeouts the queueing mechanisms are of interest.

A small example of a logging entry when sending an inquiry command to a device shows this.

```
Jun 27 08:44:58 localhost kernel: sd 5:0:0:0: [sde] sd_ioctl: disk=sde, cmd=0x2285
Jun 27 08:44:58 localhost kernel: sd 5:0:0:0: [sde] sd_ioctl: disk=sde, cmd=0x2285
Jun 27 08:44:58 localhost kernel: sd 5:0:0:0: [sde] sd_ioctl: disk=sde, cmd=0x2285
Jun 27 08:44:58 localhost kernel: sd 5:0:0:0: [sde] sd_ioctl: disk=sde, cmd=0x2285
```

10.2.2 Tracing

One of the tracing options is utilising the kernels BPF [5] [29] instrumentation to obtain detailed information of basically every kernel function available. Based on this functionality a fair few tools have been developed to aid in using tracing storage related information. The second one is by using the `ftrace` [6] kernel instrumentation and is exposed via the `/sys/kernel/debug` pseudo filesystem. The tool to use that is the `trace-cmd` utility by Steven Rodstedt who is the developer of the ftrace kernel part as well as this tool. Be aware this is worth a book on its own and would be significant detour from the purpose of this book and therefore I won't be diving further into this. If you want to know more about it please refer to the `ftrace` kernel documentation [30]

10.2.2.1 Blktrace

Blktrace uses the `ftrace` kernel function tracer by instrumenting the `blk` tracer and basically does the hard work in `ftrace` for you. The `blktrace` package comes with a few tools. The primary one is obviously `blktrace` which is capturing the IO commands going to a device and the responses returned. The output is stored in binary format so you would need to use the `blkparse` tool to interpret these.

An example:

[5] Berkley Packet Filter
[6] Function Tracer

```
sserver:~ # blktrace -d /dev/sdd -o - | blkparse -in -
  8,48   3        1     0.000000000  2887  D   R 36 [sg_inq]
  8,48   3        2     0.000091843     0  C   R [0]
  8,48   3        3     0.000105292  2887  D   R 252 [sg_inq]
  8,48   3        4     0.000166053     0  C   R [0]
  8,48   0        1    37.487601624  2889  D   R 252 [sg_vpd]
  8,48   0        2    37.487756454     0  C   R [0]
  8,48   2        1    49.721431588  2890  D   R 252 [sg_vpd]
  8,48   2        2    49.721564509     0  C   R [0]
```

As you can see you can manipulate the output of `blktrace` and pipe this directly to `blkparse`. The above shows a simple output of the major:minor, the cpu which handled the IO,sequence id, time since blktrace started, SCSI opcode, trace action, RWBS field indicating Read/Write/Discard or either a B (barrier) or S (synchronous operation)

These output can be adjusted to what you need to see. See the man page for further information.

A further utility that comes with `blktrace` is `btt`. This little tool is most useful for statistical analysis of the IO operations. It is very helpful to see which process is issuing IO's when multiple applications are using the system. You can then use the `btt` tool to extract IO's from just that application. If you use the `-B` parameter of `btt` 3 separate files will be created that also gives you the option to either use it in a spreadsheet or use the accompanied `bno_plot` utility to provide a graph.

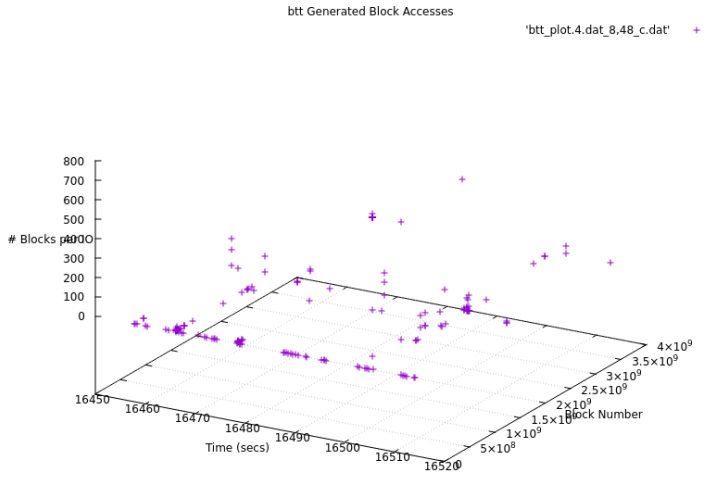


Figure 10.2: bno plot output

10.2.2.2 BCC

BCC [31] is not really a single tool but more a collection of scripts and utilities that instruments a BPF filter on a variety of kernel trace-points. It contains a fair few nifty things like `biosnoop`, `biotop`, `bitesize` and a whole lot more.

A few examples:

`biolatency` shows the distribution of IO's subject to various latencies. It does not show why this happens but gives an indication of latency the IO's observe. Various parameters adjust the behaviour and output such as `-Q` which would allow queue time to be included.

```
sserver:~ # /usr/share/bcc/tools/biolatency
Tracing block device I/O... Hit Ctrl-C to end.
^C
     usecs               : count     distribution
         0 -> 1          : 0        |                                        |
         2 -> 3          : 0        |                                        |
         4 -> 7          : 0        |                                        |
         8 -> 15         : 0        |                                        |
        16 -> 31         : 7        |***                                     |
        32 -> 63         : 14       |******                                  |
        64 -> 127        : 70       |*********************************       |
       128 -> 255        : 84       |****************************************|
       256 -> 511        : 54       |*************************               |
       512 -> 1023       : 5        |**                                      |
      1024 -> 2047       : 1        |                                        |
      2048 -> 4095       : 0        |                                        |
      4096 -> 8191       : 9        |****                                    |
sserver:~ #
```

A second example is `bitehist` which provides a histogram of the IO size distribution per process.

```
sserver:~ # /usr/share/bcc/tools/bitesize
Tracing block I/O... Hit Ctrl-C to end.
^C
Process Name = kworker/0:1H
     Kbytes              : count     distribution
         0 -> 1          : 0        |                                        |
         2 -> 3          : 0        |                                        |
         4 -> 7          : 1        |****************************************|

Process Name = kworker/0:2
     Kbytes              : count     distribution
```

```
         0 -> 1          : 3        |****************************************|

Process Name = jbd2/sda2-8
     Kbytes              : count     distribution
         0 -> 1          : 0        |                                        |
         2 -> 3          : 0        |                                        |
         4 -> 7          : 0        |                                        |
         8 -> 15         : 1        |****************************************|
```

There are a lot more of these nifty tools and my advice is to learn and test with them, so you can apply this in various troubleshooting scenarios.

You might wonder why I've put these paragraphs in the Protocols/SCSI chapter. It is true that most of these tools can be used in various locations of the IO stack and I maybe highlighting some functions of these tools in the relevant chapters in the future. The SCSI and NVMe layer is more or less the cross-section between hardware and software and can provide detailed information on IO's requests that get submitted as well as the response codes coming from the devices. It has a very good logging and tracing facility and most tools can hook into this layer for all sorts of information.

Although the above mentioned tools seem to be focussed on performance diagnostics, you have to be aware that problems can arise from performance issues as well as the other way around where either configuration and design mistakes as well as infrastructure problems can cause these performance degradation. One of the more classic examples is when the transport layer is set up to carry data subject to very different IO profiles. If an application has data sets that create very large IO's on a set of disks and very small IO's on a different set of disks, but are still served out of the same array port(s) and HBA's or network cards, you can imagine that at some stage this will clash. If a port out of a SCSI array has a target queue depth of 128 commands and 120 are already occupied by very large write IO's it will take time for these to get de-staged to either cache or disks. All the time the same server, and even others if linked to that same port, are restricted in the amount of data they can send. Very often you will then see a response code like "queue full". It is therefore imperative that the IO profiles of the applications are known and the design of the system and infrastructure is aligned to these characteristics and able to cater for these differences.

If you use the above tools properly you can determine the cause of the errors and follow the link either upstream to other parts of the IO stack in the server itself or to parts that start touching the infrastructure such as network cards, fibre channel host bus adapters etc.

10.2.3 DIF-DIX

One of the main issues that have plagued system administrators and application users is data corruption. Not much is needed for one or more bits to get flipped between the application and the physical storage medium. Although many checks and verification options have been put in place on almost every level of the IO stack there never has been a real end-to-end

solution. Oracle came pretty close with HARD but that did not store the checksums on disk so there was only a one-way verification on writes.

To be able to detect discrepancies in both read and write direction a so call DIF [7] field is added for each of the IO's aligned on a 512 byte boundary. Be aware that diskdrives where always aligned on 512 byte sector sizes. Array vendors however have started using 520 byte sector sizes internally to cater for internal data integrity checks.

So what does this entail. Basically what happens is that an 8-byte field is added to each 512-byte chunk of data for which the operating system is creating the IO. This field has a 2-byte `Guard` tag ,which is a CRC value of the 512-byte data field, a 2-byte `APP` tag, which can be used by the operating system for whatever use it deems useful and a 4 byte `REF` field which represents the lower 32-bits of the sector on disk. Depending on the support of the HBA, drivers and storage array the entire path from the HBA to the disk can then verify if that data field is actually correct and no bits have flipped anywhere along the line. As the verification field is stored with the data itself means that any future validation can be done in various ways.

Although many of the layers of the IO stack have measures in place to check on the validity of the data it handles it has never been really end-to-end. For example the transport layers themselves have a CRC or some other kind of checksum, this is however stripped from the frame/packet upon arrival at the destination and thus has no longevity for future checks. Filesystems like BTRFS have internal checksumming and will correct blocks that show errors.

Having the integrity check incorporated along the actual data means that checks can always be performed at any given point in time.

Support for the DIF field needs to be enabled on a few places.

- HBA (or network card in case of iSCSI)
- Array target port
- Disk (LUN)

On a Broadcom/Emulex HBA it can be enabled via the `lpfc_enable_bg` module parameter and `ql2xenabledif` for Marvell/Qlogic adapters. Secondly the target ports on the array also need to be aware that the DIF field needs to be enabled as any read command that is sent from the HBA is followed by the array sending that data. If the array does not support the DIF method or it is not enabled it will not add that 8-byte field. Lastly, and this really depends on the array vendor, model and firmware version, the disks presented out of the array need to be formatted with DIF enabled as well as it would need to cater for the additional 8-bytes on each sector. Even if the array already natively formats the disks with a 520 byte sector size it would still need to change the method from any internal proprietary way to the SCSI T10 DIF standard. Again verify with your array vendor what is required on that side.

[7] Data Integrity Field AKA T10 PI (Protection Information)

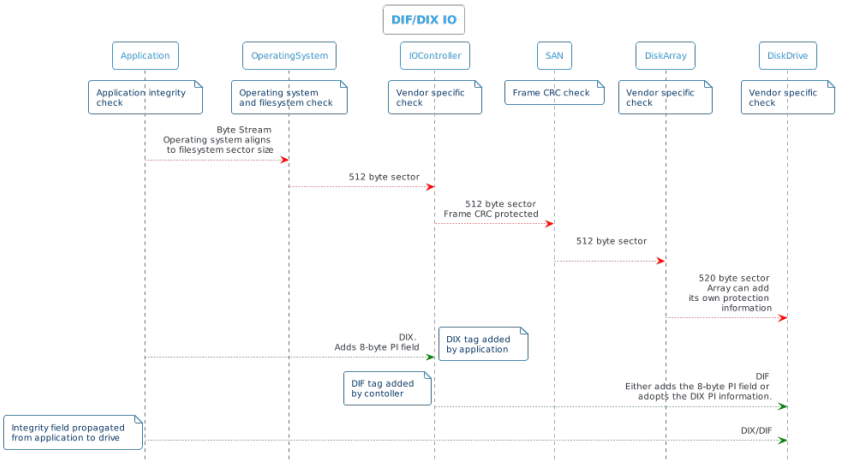


Figure 10.3: DIF/DIX IO

If everything is configured properly you should see something like this from the `lsscsi -p -P` output

```
#### lsscsi -p -P -x -s ########################

[11:0:0:0x0000]              disk    HITACHI  OPEN-V  \
        8801  /dev/sdc   DIF/Type1  T10-DIF-TYPE1-CRC  dix1  10.7GB
[11:0:0:0x0001]              disk    HITACHI  OPEN-V  \
        8801  /dev/sdd   DIF/Type1  T10-DIF-TYPE1-CRC  dix1  10.7GB
```

The T10 [8] has no saying in how operating systems should handle OS to controller communications. That is all handled via standards outside of its scope. The SCSI standard does define how IO commands can and should be executed in order to get things done on the end device. It takes the IO request from the OS and creates a so called CDB [9]. That CDB contains the instruction set so the remote device knows what to do.

Depending on the support of the remote device it determines if a CDB [10] should contain a RDPROTECT or WRPROTECT field populated with the type of information it requires as per SCSI standard. That instruction is then sent on to the IO controller with a further instruction of what type of IO to perform.

The mode of supported operations is defined as follows.

[8] SCSI standards committee
[9] Command Descriptor Block
[10] Command Descriptor Block

Table 10.1: T10 PI Protection Modes

Mode	Description
T10 Type 0	Normal, unprotected I/O.
T10 Type 1	Protection supported between controller and a storage device formatted with T10 Type 1 protection information.
T10 Type 2	Protection supported between controller and a storage device formatted with T10 Type 2 protection information.
T10 Type 3	Protection supported between controller and a storage device formatted with T10 Type 3 protection information.
DIX Type 0	Protection DMA enabled between OS and controller. Storage device is not formatted with protection information.
DIX Type 1	Protection DMA enabled between OS and controller. Storage device is formatted with T10 Type 1.
DIX Type 2	Protection DMA enabled between OS and controller. Storage device is formatted with T10 Type 2.
DIX Type 3	Protection DMA enabled between OS and controller. Storage device is formatted with T10 Type 3.

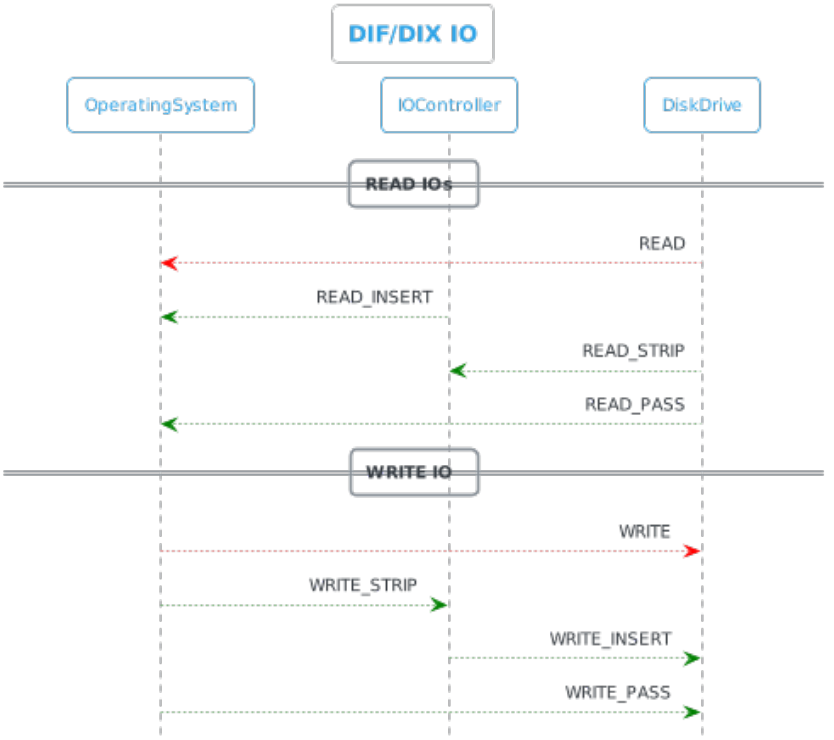


Figure 10.4: DIF/DIX IO Type

- T10 Types 1-3 indicate that the controller implements the READ_STRIP and WRITE_INSERT operations.

- DIX Type 0 indicates that the controller implements the READ_INSERT and WRITE_STRIP operations.
- DIX Types 1-3 indicate that the controller implements the READ_PASS and WRITE_PASS operations.

As you can see in fig. 10.4 the interaction between the various layers is determined based on what is supported from an application layer to the actual drive. Refer to the documentation of what is supported on each of the layers of the IO stack.

When either of the supported modes (DIX and/or DIF) is enabled and the verification check fails the operating system return a SCSI error with a status check condition.

Table 10.2: T10 Block Guard ASC/ASCQ status codes.

ASC/ASCQ code	Description
10h/00h	id crc or ecc error
10h/01h	logical block guard check failed
10h/02h	logical block application tag check failed
10h/03h	logical block reference tag check failed
10h/04h	logical block protection error on recover buffered data
10h/05h	logical block protection method error

These error codes will be logged in the various logging facilities you have configured.

10.3 NVMe

With the advance of Solid State Drives in the early 2010's, the inherent limitations of the SCSI protocol became more and more apparent. As I mentioned before SCSI was designed and developed over a period of over 30 years so there is a lot of `fat` (i.e. overhead) which is not needed for device communication to store and retrieve data. It's single queuing mechanism has never been a problem when it needed to server mechanical disk drives (HDD's) as the delay imposed by these drives could never result in an actual bottleneck in the host's IO stack. SSD's on the other hand have no such thing as mechanical delay and IO commands are instantly served by the electronics based devices which behave more like memory. NVMe grew out of the PCI Express world and was build from the ground up to utilise the the characteristics from SSD in the most optimal way.

If you want to know more about NVMe from a protocol perspective I would advise to visit the NVMe [27] website. There you will find many resources including presentations, video's as well as the full specification.

For communicating with NVMe devices on a Linux platform we use the `nvme` cli command. Unless NVMe devices are not directly attached to the host system the main cause for errors will almost always be hardware related. NVMe has two methods of communication, either memory of message methods are used. Devices that are directly attached to the host system PCI bus use a memory based method and devices that are mapped through an transport controller, for example an NIC card in case of TCP transport a FC Host Bus Adapter in case of NVNe over Fabrics. A third option is when controllers can use either such as RDMA controllers. From an administrative standpoint you only need to know the namespace you're dealing with as the `nvme` cli command will just talk to the NVMe driver who will abstract these commands for you and utilise the underlying communications method. It is however important you know the layout of your system. A host can have multiple NVMe controllers who can each server a large number of namespaces.

There are a number of `nvme` cli command parameters that can help you out here. If you see IO errors in the Linux message log it will always be accompanied with the disk itself in the same way this happens for SCSI based devices.

The first thing you want to know is the actual state of the device itself and if any hardware issues are seen. The `nvme smart-log` command helps you out here.

```
[root@server:~]$ sudo nvme smart-log /dev/nvme0n1
Smart Log for NVME device:nvme0n1 namespace-id:ffffffff
critical_warning 0
temperature                 : 36 C
available_spare             : 100%
available_spare_threshold   : 10%
percentage_used             : 0%
endurance group critical warning summary: 0
data_units_read             : 1,158,911
data_units_written          : 3,365,890
host_read_commands          : 22,702,286
host_write_commands         : 29,609,766
controller_busy_time        : 198
power_cycles                : 300
power_on_hours              : 97
unsafe_shutdowns            : 8
media_errors                : 0
num_err_log_entries         : 193
Warning Temperature Time    : 0
Critical Composite Temperature Time : 0
Temperature Sensor 1           : 36 C
Temperature Sensor 2           : 39 C
Thermal Management T1 Trans Count : 0
Thermal Management T2 Trans Count : 0
Thermal Management T1 Total Time : 0
Thermal Management T2 Total Time : 0
```

In the above output you see that there are no media errors or any other significant issues that require attention. The `num_error_log_entries` field does show there are 193 entries so lets have a look at that.

```
[root@server:~]$ nvme error-log /dev/nvme0
Error Log Entries for device:nvme0 entries:64
.................
 Entry[ 0]
.................
error_count : 195
```

```
sqid  : 0
cmdid  : 0x2
status_field : 0x4212(INVALID_LOG_PAGE: The log page indicated is invalid. \
  This error condition is also returned if a reserved log page is requested)
parm_err_loc : 0x28
lba  : 0
nsid  : 0xffffffff
vs  : 0
trtype  : The transport type is not indicated or the error is not transport related.
cs  : 0
trtype_spec_info: 0
.................

---snip---

.................
 Entry[63]
.................
error_count : 0
sqid  : 0
cmdid  : 0
status_field : 0(SUCCESS: The command completed successfully)
parm_err_loc : 0
lba  : 0
nsid  : 0
vs  : 0
trtype  : The transport type is not indicated or the error is not transport related.
cs  : 0
trtype_spec_info: 0
```

In the disk that I have here there is space for 64 entries in the error log. As it is a circular log the oldest will be overwritten by new entries. The messages are somewhat cryptic and you would need to refer back to the NVMe specification to find out what these mean and refer to. As you can see even though the terminology says `error log` it does not alway have to refer to actual media errors. It also logs entries as a result of non-supported commands being send to the controller or device.
Another thing to note here is that the messages are not time stamped so it is often very difficult to correlate events in the linux messages log to an entry in the device error log.

The last part I want to highlight here related to NVMe is support for certain functions and feature you may need. An example is end-to-end data integrity verification which you may have come across from the SCSI world. To check if this feature is supported on your hardware you use the `nvme get-feature /dev/nvme0 -n 1 -f 0x16 -s 0 -H` command. There

are numerous features you can query by changing the hex number after the `-f` parameter however the actual reporting depends if your hardware supports the feature or not.

An example of a useful command is determining the number of IO submission and completion queues.

```
 root@server ~: nvme get-feature /dev/nvme0n1 -n 1 -f 0x7 -H
get-feature:0x7 (Number of Queues), Current value:0x1f001f
  Number of IO Completion Queues Allocated (NCQA): 32
  Number of IO Submission Queues Allocated (NSQA): 32
```

Be aware that vendors add more and more functionality in their hardware not only with newer generation systems but also firmware. Look at the release notes of the firmware the vendor publishes and determine if this is useful. (Don't determine the usefulness on this though. In many cases significant improvements are made overall)

10.4 Transport

The transport protocol is separated from the command and control layer as it used to be when devices where more directly attached to the host system itself. For obvious reasons this did not scale well and storage became more network oriented. This allowed for more flexibility, as well as performance improvements, as dedicated storage controllers could now be used to offload commands without the host IO needed to be tasked with handling storage maintenance tasks. In the earlier days when an application needed to write a block of data the OS not only needed to setup the stage for the data to be sent to the device but also had to arrange communication with the device firmware to actually have the correct setup of the physical spindles like driving the actuators and waiting for completion etc. Each of these causes the host processor to stop serving applications as it needs to act on the various interrupts which obviously incurred a performance impact. On high end systems attached to SAN environments or hardware raid controllers that is not something you need to worry anymore.

The abstraction of the device communication protocol and transport protocol did however increase the complexity significantly. Instead of simply opening up a server and check the internal controller and disks you now had to look at an infrastructure that provided that functionality.

Providing storage facilities over a network was not new. SUN [32] Microsystems already provided us with NFS [33] (Network File System) and Microsoft gave us SMB (Server Message Block) and each of which have developed through various iterations over the years.

These systems did however not provide block level storage in the sense that applications could store data directly. There always was a network software abstraction layer in between.

10.4.1 iSCSI

iSCSI is a command and control abstraction layer that provides block level addressable storage space. The underlying transport is TCP/IP. From a design perspective TCP/IP has a lossy networked state of mind. Inherently this means that there is a lot of background code that takes care of transmission control, retransmission, slow-starts, bandwidth correction via adjusting TCP window sizes etc. If a network is well designed and the proper measures are in place so that the iSCSI traffic is not interrupted or subject to bandwidth restrictions potentially due to shared links it may be a very good solution for many environments.

The OpeniSCSI [34] implementation, most often used in Linux distributions, contains a GPL licensed kernel part, which takes care of all the effective data transfer, and a userland part that handles the entire control plane from configuration to discovery and mapping etc.

Below is a small diagram of a network setup I use to demonstrate and highlight issues that can occur in an iSCSI environment. The diagram already outlines the concept of access path separation for both the iscsi initiator to target traffic as well as client to host traffic to the iscsi iniator.

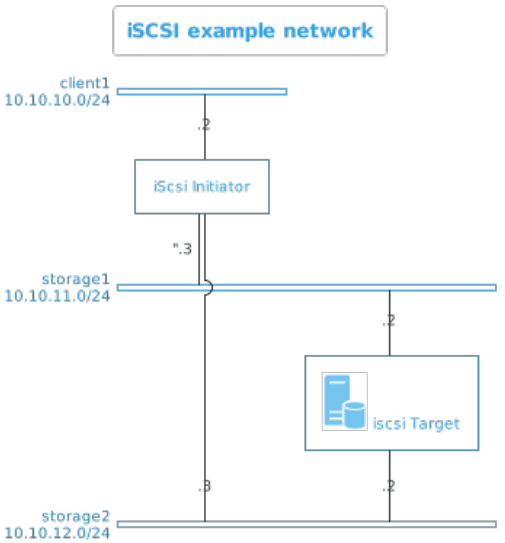


Figure 10.5: Sample iSCSI network

The diagram should be interpreted as having three physically separate networks. It would not make much sense if these would represent one physical link carved up in a few vlan's

traversing the corresponding IP subnets. Although a few things can be accomplished that way like QoS [35] it will not prevent any redundancy if any physical issues would occur on the link.

By having an air-gap between these networks you ensure at least two things:

1. Client traffic does not interfere with initiator to target traffic.
   - This ensures a much better control over network flow characteristics.
2. Redundant access paths from the initiator to the target are safeguarded.
   - If any thing would happen on a network layer in one network the initiator would still have access to the target via the other network.

The same concepts are also applied on FibreChannel networks as you will see below and in the MPIO chapter.

What this does not guarantee is a slow-drain propagation of one network onto another. What I mean by that is when problems occur on one network this will have an impact in the IO behaviour on the host. This can, and often will, result in the fact that commands executed on the other network may need to wait for earlier commands to be completed. This may not be driven by the IO stack itself but most applications want their IO's completed sequentially in order to safeguard consistency and integrity.
So even when there is an air-gap in between networks or fabrics, a host or target observing issues in one network may often be propagating actual IO delay onto another. We will see in the Fibre Channel section how this can be alleviated by some new technology that has made its way into the FC standards.

The iSCSI protocol is not born out of the INCITS T10 committee but adheres to those various standards such as SBC [11], SPC [12] etc. These standards are all defined in the SCSI Architecture [36] model. iSCSI is based on RFC 7134 [37] and is very well documented. I will not outline the protocols in-depth here as the RFC is very clear.

Most, if not all, Linux distributions provide the option to install the iSCSI initiator, target or both. The target provides the source disks that will be exposed to the clients. This already means that there are a few thing that need to be setup properly.

- The disks that will be exposed to the clients. These can be sourced via various methods.
  - A local disk in the host system itself.
  - A file
  - A ram disk
  - A passthrough device
  - Even a disk provisioned out of a SAN.
- Client access
  - Network interfaces
  - TCP port usage

[11] SCSI Block Commands
[12] SCSI Primary Commands

  - Access Control
  - Authentication
  - Mapping
- Function control and protocol support
  - ALUA
  - Persistent Reservation

10.4.1.1 targetcli

The `targetcli` utility is a wrapper tool to dynamically configure the target. It is not installed by default so a `dnf install targetcli` will get that sorted. On SUSE systems you can use `yast` or `zypper` to achieve the same.

```
opensuse:~ # targetcli
targetcli shell version 2.1.53
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/> ls
o- / ......................................................................... [...]
  o- backstores .............................................................. [...]
  | o- block .................................... [Storage Objects: 1]
  | | o- idiskc ..................... [/dev/sdc (8.0GiB) write-thru activated]
  | |   o- alua ....................................... [ALUA Groups: 1]
  | |     o- default_tg_pt_gp ................... [ALUA state: Active/optimized]
  | o- fileio ................................... [Storage Objects: 0]
  | o- pscsi .................................... [Storage Objects: 0]
  | o- ramdisk .................................. [Storage Objects: 0]
  o- iscsi ............................................ [Targets: 2]
  | o- iqn.2003-01.org.linux-iscsi.opensuse.x8664:sn.4bcc3c8887b4 .... [TPGs: 1]
  | | o- tpg1 ..................................... [no-gen-acls, no-auth]
  | |   o- acls ................................................ [ACLs: 0]
  | |   o- luns ................................................ [LUNs: 1]
  | |   | o- lun0 ................. [block/idiskc (/dev/sdc) (default_tg_pt_gp)]
  | |   o- portals ............................................ [Portals: 1]
  | |     o- 10.10.12.2:3260 ......................................... [OK]
  | o- iqn.2003-01.org.linux-iscsi.opensuse.x8664:sn.cfb18c8076f7 .... [TPGs: 1]
  |   o- tpg1 ..................................... [no-gen-acls, no-auth]
  |     o- acls ................................................ [ACLs: 0]
  |     o- luns ................................................ [LUNs: 1]
  |     | o- lun0 ................. [block/idiskc (/dev/sdc) (default_tg_pt_gp)]
  |     o- portals ............................................ [Portals: 1]
  |       o- 10.10.11.2:3260 ......................................... [OK]
```

```
  o- loopback ..................................................... [Targets: 0]
  o- vhost ........................................................ [Targets: 0]
  o- xen-pvscsi ................................................... [Targets: 0]
/>iqn.2003-01.org.linux-iscsi.opensuse.x8664:sn.cfb18c8076f7
```

10.4.1.2 iSCSI naming

In the output above you see a fairly long string starting with `iqn` which stands for `iSCSI Qualified Name` [38]. The RFC mentions three naming options but the `iqn` is most often used.
It consists of three or four parts.

- The string `iqn`. To ensure no ambiguity with other names.
- A date mark representing the year and month the naming authority owned the domain name.
- The domain name in reverse notation.
- And optionally a semicolon followed by a string which may contain some device identifier or other info the administrator deems required/useful.

In the example above you can see the server has one disk configured to be exposed. (/dev/sdc as idiskc). In the `iscsi` section two targets have been defined, `iqn.2003-01.org.linux-iscsi.opensuse.x8664:sn.4bcc3c8887b4` and `iqn.2003-01.org.linux-iscsi.opensuse.x8664:sn.cfb18c8076f7`. Each of them have a target port group with the ACL's, luns, IP addresses and tcp ports set.

In this scenario every iscsi client can connect to these targets and that `idiskc` lun will be presented. That is obviously not what you want. In order to prevent cross mapping between clients and as such risking all sorts of nastiness resulting in IO errors, filesystem and data-corruption, we need to configure an ACL on each target port-group.

```
/iscsi/iqn.20...7b4/tpg1/acls> create iqn.2021-04.com.example:01:3855b8fbcc15
Created Node ACL for iqn.2021-04.com.example:01:3855b8fbcc15
Created mapped LUN 0.
/iscsi/iqn.20...6f7/tpg1/acls> ls /
o- / ......................................................................... [...]
  o- backstores .............................................................. [...]
  | o- block .................................... [Storage Objects: 1]
  | | o- idiskc ..................... [/dev/sdc (8.0GiB) write-thru activated]
  | |   o- alua ....................................... [ALUA Groups: 1]
  | |     o- default_tg_pt_gp .................. [ALUA state: Active/optimized]
  | o- fileio ................................... [Storage Objects: 0]
  | o- pscsi .................................... [Storage Objects: 0]
  | o- ramdisk .................................. [Storage Objects: 0]
  o- iscsi ............................................... [Targets: 2]
  | o- iqn.2003-01.org.linux-iscsi.opensuse.x8664:sn.4bcc3c8887b4 ... [TPGs: 1]
```

```
  | | o- tpg1 ..................................... [no-gen-acls, no-auth]
  | |   o- acls ................................................ [ACLs: 1]
  | |   | o- iqn.2021-04.com.example:01:3855b8fbcc15 ......... [Mapped LUNs: 1]
  | |   |   o- mapped_lun0 ........................ [lun0 block/idiskc (rw)]
  | |   o- luns ................................................ [LUNs: 1]
  | |   | o- lun0 ............... [block/idiskc (/dev/sdc) (default_tg_pt_gp)]
  | |   o- portals .............................................. [Portals: 1]
  | |     o- 10.10.12.2:3260 ........................................... [OK]
  | o- iqn.2003-01.org.linux-iscsi.opensuse.x8664:sn.cfb18c8076f7 ... [TPGs: 1]
  |   o- tpg1 ..................................... [no-gen-acls, no-auth]
  |     o- acls ................................................ [ACLs: 1]
  |     | o- iqn.2021-04.com.example:01:3855b8fbcc15 ......... [Mapped LUNs: 1]
  |     |   o- mapped_lun0 ........................ [lun0 block/idiskc (rw)]
  |     o- luns ................................................ [LUNs: 1]
  |     | o- lun0 ............... [block/idiskc (/dev/sdc) (default_tg_pt_gp)]
  |     o- portals .............................................. [Portals: 1]
  |       o- 10.10.11.2:3260 ......................................... [OK]
  o- loopback ................................................... [Targets: 0]
  o- vhost ...................................................... [Targets: 0]
  o- xen-pvscsi ................................................. [Targets: 0]
/iscsi/iqn.20...6f7/tpg1/acls>
```

The `iqn` now showing under the `acls` is the one that is inialised on the client pseudo-randomly. As you can seen the `idiskc` disk is automatically mapped. That is also a setting you need to change as it will lead to issues with subsequent mappings of client to the same portal.

The initiator is merely doing a discovery and when the target allows the access the disks will be provisioned. There is not very much you can do or see from a client side apart from providing the correct information and starting the iSCSI session to the target.

When the discovery and assignments have been correctly done you can verify with the `dmesg` output what its outcome is.

```
<snip>
[   17.844939] Loading iSCSI transport class v2.0-870.
[   17.870745] iscsi: registered transport (tcp)
[   17.873625] scsi host3: iSCSI Initiator over TCP/IP
[   17.880274] scsi 3:0:0:0: Direct-Access     LIO-ORG  idiskc           4.0
[   17.881945] scsi 3:0:0:0: alua: supports implicit and explicit TPGS
[   17.881948] scsi 3:0:0:0: alua: device naa.60014056a02beea37514b0084cffc3ab....
[   17.882073] scsi 3:0:0:0: Attached scsi generic sg2 type 0
[   17.887962] sd 3:0:0:0: [sdb] 16777216 512-byte logical blocks: (8.59 GB/8.00 GiB)
[   17.888344] sd 3:0:0:0: [sdb] Write Protect is off
[   17.888345] sd 3:0:0:0: [sdb] Mode Sense: 43 00 00 08
```

```
[   17.889222] sd 3:0:0:0: [sdb] Write cache: disabled, read cache: enabled.....
[   17.890177] sd 3:0:0:0: [sdb] Optimal transfer size 4194304 bytes
[   17.903755] sd 3:0:0:0: alua: transition timeout set to 60 seconds
[   17.903781] sd 3:0:0:0: alua: port group 00 state A non-preferred supports TOlUSNA
[   17.928820] sd 3:0:0:0: [sdb] Attached SCSI disk
[   18.028060] device-mapper: multipath service-time: version 0.3.0 loaded
[   18.115586] sd 3:0:0:0: alua: port group 00 state A non-preferred supports TOlUSNA
<snip>
[ 1674.876348] scsi host4: iSCSI Initiator over TCP/IP
[ 1674.881094] scsi 4:0:0:0: Direct-Access     LIO-ORG  idiskc           4.0
[ 1674.882787] scsi 4:0:0:0: alua: supports implicit and explicit TPGS
[ 1674.882791] scsi 4:0:0:0: alua: device naa.60014056a02beea37514b0084cffc3ab.....
[ 1674.883427] sd 4:0:0:0: Attached scsi generic sg3 type 0
[ 1674.887534] sd 4:0:0:0: [sdc] 16777216 512-byte logical blocks: (8.59 GB/8.00 GiB)
[ 1674.891235] sd 4:0:0:0: [sdc] Write Protect is off
[ 1674.891238] sd 4:0:0:0: [sdc] Mode Sense: 43 00 00 08
[ 1674.892463] sd 4:0:0:0: [sdc] Write cache: disabled, read cache: enabled.....
[ 1674.895859] sd 4:0:0:0: [sdc] Optimal transfer size 4194304 bytes
[ 1674.910145] sd 4:0:0:0: alua: port group 00 state A non-preferred supports TOlUSNA
[ 1674.983364] sd 4:0:0:0: [sdc] Attached SCSI disk
[ 1675.073917] sd 3:0:0:0: alua: port group 00 state A non-preferred supports TOlUSNA
<snip>
```

10.4.1.3 Problem determination

As mentioned before the majority of O issues in most storage environments is coming from the transport layer as that is the most error-prone. In case of iSCSI this is TCP/IP. Let me rephrase that, TCP/IP is not error-prone, the underlying infrastructure as well as many network designs are simply not suitable to carry a storage based protocol. iSCSI is not alone in this. We'll touch on this later with Fibre Channel over IP (FCIP)

So in order to troubleshoot an iSCSI based storage path we need to determine where the issue is actually originating. One of the most common issues is that you simply are not able to see the provisioned disk. To start here first check and double check the target configuration. Are all disks mapped, are the target port groups set up correctly, are the access control lists not prohibiting the iSCSI initiator from actually accessing the target, is the iSCSI service actually running.

By default some distributions do require to manually enable the iSCSI service. An easy check:

```
opensuse:~ # systemctl status iscsi

• iscsi.service - Login and scanning of iSCSI devices
```

```
     Loaded: loaded (/usr/lib/systemd/system/iscsi.service; enabled; vendor preset: enabled)
     Active: active (exited) since Fri 2022-03-18 10:58:18 AEST; 1h 29min ago
       Docs: man:iscsiadm(8)
             man:iscsid(8)
    Process: 1237 ExecStart=/usr/sbin/iscsiadm -m node --loginall=automatic \
      -W (code=exited, status=21)
    Process: 1238 ExecStart=/usr/sbin/iscsiadm -m node --loginall=onboot \
      -W (code=exited, status=21)
    Process: 1239 ExecStart=/usr/sbin/iscsiadm -m fw -l -W (code=exited, status=21)
   Main PID: 1239 (code=exited, status=21)
        CPU: 7ms

Mar 18 10:58:18 opensuse systemd[1]: Starting Login and scanning of iSCSI devices...
Mar 18 10:58:18 opensuse iscsiadm[1237]: iscsiadm: No records found
Mar 18 10:58:18 opensuse iscsiadm[1238]: iscsiadm: No records found
Mar 18 10:58:18 opensuse iscsiadm[1239]: iscsiadm: Could not get list of targets from \
   firmware. (err 21)
Mar 18 10:58:18 opensuse systemd[1]: Finished Login and scanning of iSCSI devices.
```

As you've seen in the example above there were two portals created by the `targetcli` tool. Each portal is listening on its own socket. A simple `netstat` will show this as follows:

```
opensuse:~ # netstat -altwn
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 10.10.11.2:3260         0.0.0.0:*               LISTEN
tcp        0      0 10.10.12.2:3260         0.0.0.0:*               LISTEN
```

The next thing to verify is if your firewall settings are correct and do not prevent valid iSCSI clients to access the above mentioned ports. As various distros use different firewall solutions I will refer to the documentation for those solutions.

Proceeding with checks is to see if there are any other low layer network issues that may prohibit a proper connection to the network interface of the target. The easiest way is a simple ping to the IP address but that does not guarantee everything. It is fairly common to have ICMP request blocked on a L2 or L3 network layer as it sometimes is perceived as a security issue.

A fairly quick way to establish if connection to the target over TCP port 3260 is possible is to use `telnet` although this is mostly no longer installed on Linux distributions.

```
linux:~ # telnet 10.10.11.2 3260
Trying 10.10.11.2...
Connected to 10.10.11.2.
Escape character is '^]'.
```

```
^]
```

Just the `Connected to .....` line shows that communications from the initiator to the target is possible from a network perspective. Other option may be to use a portscanner like `nmap`.

```
linux:~ # nmap -p 3260 10.10.11.2
Starting Nmap 7.91 ( https://nmap.org ) at 2021-06-01 16:53 AEST
Nmap scan report for 10.10.11.2
Host is up (0.00030s latency).

PORT     STATE SERVICE
3260/tcp open  iscsi
MAC Address: 0A:00:27:00:00:03 (Unknown)

Nmap done: 1 IP address (1 host up) scanned in 1.23 seconds
```

As network layer connectivity has now been established and we know the iscsi target is listening the initiator should now be able to connect and the provisioned disk should show up on the client.

I mention before that when a lun was created it would be automatically mapped to the respective portals. That would cause an immediate access to initiators which you don't want. In order to prevent that turn this off in the `targetcli` tool.

```
/>
/> set global auto_add_mapped_luns=false
Parameter auto_add_mapped_luns is now 'false'.

/> get global auto_add_mapped_luns
auto_add_mapped_luns=false

/> saveconfig
Configuration saved to /etc/target/saveconfig.json
```

From a client perspective there are two ways to connect to the iSCSI target.

- The first one is to directly do a discovery towards the target and
- secondly use a iSNS server.

If you have a fairly dynamic iSCSI environment where initiators, targets, portals and disks are being added and removed on a regular basis, a iSNS [39] server comes in very useful. Without going into to too much details the iSNS service provides a central repository for discovery and management purposes for both iSCSI and iFCP. An iSNS service can take a lot of the hassle out of managing iSCSI registrations and state changes. A conceptual discovery dialog how an iSNS server acts between client and server is shown below.

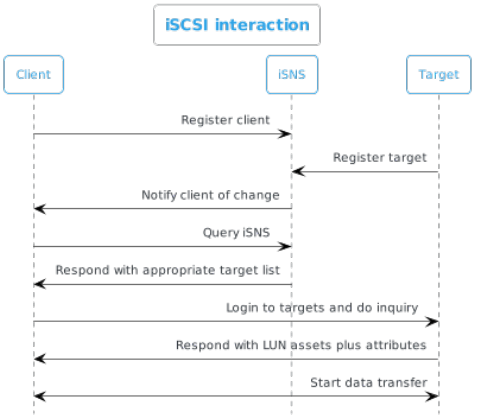


Figure 10.6: iSCSI iSNS Interaction

As the initial iSCSI configuration is set now we should be able to see the disk showing up on the initiator side.

```
linux:~ # ll /dev/sd*
brw-rw---- 1 root disk 8,  0 Jun  2 14:19 /dev/sda
brw-rw---- 1 root disk 8,  1 Jun  2 14:19 /dev/sda1
brw-rw---- 1 root disk 8,  2 Jun  2 14:19 /dev/sda2
brw-rw---- 1 root disk 8, 16 Jun  2 14:19 /dev/sdb
brw-rw---- 1 root disk 8, 32 Jun  2 14:54 /dev/sdc
```

Hmm, that's weird. There is only one disk provisioned out of the iSCSI target but I see two new popping up.

Well, remember that we have presented the same disk on the target side via two portals. The initiator is therefore presenting not two disks but merily the access paths to that disk and these are then sequentially presented to the block layer and hence you will see two disks.

When looking a bit more closely and when we start querying the disks to obtain their characteristics you will see that the client knows its the same disk

```
linux:~ # lsblk
NAME                                 MAJ:MIN RM  SIZE RO TYPE  MOUNTPOINT
sda                                    8:0    0    8G  0 disk
├─sda1                                 8:1    0    8M  0 part
└─sda2                                 8:2    0    8G  0 part  /
sdb                                    8:16   0    8G  0 disk
└─360014056a02beea37514b0084cffc3ab 254:0    0    8G  0 mpath
```

```
sdc                                     8:32   0    8G  0 disk
└─360014056a02beea37514b0084cffc3ab 254:0    0    8G  0 mpath
```

The type of the disk and the same UUID already provides a hint here.

As I indicated earlier the iSCSI protocol relies heavily on the network layer. When the administrative configuration options as described above appear to be OK the lower layers in the OSI stack should be investigated.

The easiest way to start is to look at the underlying design of the network infrastructure itself. The main objective here is to review the location of the initiator and targets and the planned flow of the data over the network. The communication of the iSNS server and the clients and target is more of a control plane and as such not really reliant on performance. That being said, delays in registration and notification can have a negative impact on path failover mechanisms and as such can cause additional delays in IO path failover and failback scenarios.

10.4.1.4 Network Considerations

As for the data flow between the initiator and target there are a few rules of thumb:

- Keep the hop count as small as possible.
  - That means that traversing physical and logical hops should be reduced to an absolute minimum.
  - Keeping them in the same physical switch in the same VLAN should be a design principal and not an afterthought in troubleshooting scenarios.
- Do not share links with other applications.
  - As sharing links will often cause forms of congestion there is a high potential of packets being dropped which have to be re-sent by the initiator or target and this will have a significant impact on performance and stability of the applications.
  - The control of shared links is very susceptible to the behaviour of other applications. If databases start doing warehousing jobs you can imagine that bandwidth will be fairly limited for other applications.
- If there is no option of isolating iSCSI datapaths with other traffic then the iSCSI traffic should be prioritised over other applications.
  - Quality of Service algorithms like L2QOS or DSCP should assist in this.

Remember that channel based protocols have been designed from the ground up to use a very reliable links between initiator and target. This started back in the 1960-ties and 1970-ties and that philosophy has not changed.

The lowest level where troubleshooting is useful is on the ethernet layer. Ensure you get cooperation of your network team and have them involved from the start. From a Linux side you are often depending on the NIC and the drivers that come with it.

A few things to look at on the ethernet side:

- NIC speed settings. (Duhh..., although you don't want to know how many cases I handled where this was set incorrectly)
- Duplex setting. (Yeah yeah. .... sigh.)
- Basically the above should not be touched and should be left to auto-negotiation. There are more link checks done and features enabled such as pause frames, DCE etc on 1G/10G and above. From a performance perspective anything less than 10G should not even be considered.
- Jumbo frames!!!! Yes 4 exclamation marks. The use of jumbo frames will significantly improve high throughput workloads. It also results in much less interrupts being fired at the CPU's and a much more efficient handling from NIC to memory is possible. Reassembly of frames into larger IO requests from the initiator or target is also reduced. Work with your network team to have this configured correctly.
- Ethtool is your friend. It's the swiss army knife for NIC configuration settings. Check the NIC user guide if `ethtool` is supported as some vendors use their own proprietary tools. Learn to use `ethtool`!! Not just executing the command but also how to interpret the output as well as setting the correct parameters. As mentioned these are most often tied to the NIC driver.
- Know your hardware! Setting the correct options will make a world of difference.
  - Proper settings related to send and receive buffers, interrupt pinning, ntuple settins, SRIOV etc need to be properly set in order to pull the maximum out of the hardware.

The first thing to know it to verify the network card you're dealing with and its capabilies.

In the below example you see two dual port Intel XGBE nics

```
[admin@server ~]$ lspci | grep net
01:00.0 Ethernet controller: Intel Corporation Ethernet Controller \
  10-Gigabit X540-AT2 (rev 01)
01:00.1 Ethernet controller: Intel Corporation Ethernet Controller \
  10-Gigabit X540-AT2 (rev 01)
04:00.0 Ethernet controller: Intel Corporation Ethernet Controller \
  10-Gigabit X540-AT2 (rev 01)
04:00.1 Ethernet controller: Intel Corporation Ethernet Controller \
  10-Gigabit X540-AT2 (rev 01)
```

The driver version shows a few options that can be used for this particular card.

```
[admin@server ~]$ modinfo ixgbe
filename:       /lib/modules/4.18.0-240.22.1.el8_3.x86_64/kernel/drivers/net/ \
  ethernet/intel/ixgbe/ixgbe.ko.xz
version:        5.1.0-k-rh8.2.0
license:        GPL v2
description:    Intel(R) 10 Gigabit PCI Express Network Driver
author:         Intel Corporation, <linux.nics@intel.com>
```

```
rhelversion:    8.3
srcversion:     525452BB6E4B60467875FAD
<snip>
parm:           max_vfs:Maximum number of virtual functions to allocate \
  per physical function - default is zero and maximum value is 63. (Deprecated) (uint)
parm:           allow_unsupported_sfp:Allow unsupported and untested SFP+ modules on \
  82599-based adapters (uint)
parm:           debug:Debug level (0=none,...,16=all) (int)
```

The `ethtool -k <nic>` shows the current features that these cards and drivers expose. Be aware that your card may show different options with different values. Anything you can use as an offload option by the NIC may be beneficial. Be aware I say "may be" as in some cases there can be some drawback especially when troubleshooting issues where underlying NIC driver issues may hide things that could be of significance.

```
[admin@server ~]$ ethtool -k eno1
Features for eno1:
rx-checksumming: on
tx-checksumming: on
  tx-checksum-ipv4: off [fixed]
  tx-checksum-ip-generic: on
  tx-checksum-ipv6: off [fixed]
  tx-checksum-fcoe-crc: off [fixed]
  tx-checksum-sctp: on
scatter-gather: on
  tx-scatter-gather: on
  tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: on
  tx-tcp-segmentation: on
  tx-tcp-ecn-segmentation: off [fixed]
  tx-tcp-mangleid-segmentation: off
  tx-tcp6-segmentation: on
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
rx-vlan-offload: on
tx-vlan-offload: on
ntuple-filters: off
receive-hashing: on
highdma: on [fixed]
rx-vlan-filter: on
vlan-challenged: off [fixed]
tx-lockless: off [fixed]
```

```
netns-local: off [fixed]
tx-gso-robust: off [fixed]
tx-fcoe-segmentation: off [fixed]
tx-gre-segmentation: on
tx-gre-csum-segmentation: on
tx-ipxip4-segmentation: on
tx-ipxip6-segmentation: on
tx-udp_tnl-segmentation: on
tx-udp_tnl-csum-segmentation: on
tx-gso-partial: on
tx-sctp-segmentation: off [fixed]
tx-esp-segmentation: on
tx-udp-segmentation: on
tls-hw-rx-offload: off [fixed]
fcoe-mtu: off [fixed]
tx-nocache-copy: off
loopback: off [fixed]
rx-fcs: off [fixed]
rx-all: off
tx-vlan-stag-hw-insert: off [fixed]
rx-vlan-stag-hw-parse: off [fixed]
rx-vlan-stag-filter: off [fixed]
l2-fwd-offload: off
hw-tc-offload: off
esp-hw-offload: on
esp-tx-csum-hw-offload: on
rx-udp_tunnel-port-offload: on
tls-hw-tx-offload: off [fixed]
rx-gro-hw: off [fixed]
tls-hw-record: off [fixed]
```

Pausing frames may help in situations where the amount of incoming or outgoing traffic is potentially causing buffering issues in the NIC. The NIC may instruct the network switch to hold off on sending frames. Especially in storage related traffic like iSCSI or even NFS this may be very beneficial. Both the driver as well as the network switches need to support it. When using FCoE it is even a requirement.

```
[admin@server ~]$ ethtool -a eno1
Pause parameters for eno1:
Autonegotiate: on
RX:  off
TX:  off
```

Ring buffers are used in many different kinds of data transfer devices and are used for storing

incoming and outgoing frames until they can get offloaded by either the network switch or Operating System. The ring buffers are mostly a single memory area on the card split into two sections, one for the rx side and one for the tx side. Ethtool provides the option to adjust the percentage of what each holds. This allows for assigning the correct value for a particular workload type. Some testing may needs to be done to get to the correct balance.

```
[admin@server ~]$ ethtool -g eno1
Ring parameters for eno1:
Pre-set maximums:
RX:    4096
RX Mini:  0
RX Jumbo:  0
TX:    4096
Current hardware settings:
RX:    512
RX Mini:  0
RX Jumbo:  0
TX:    512
```

Interrupt rate limiting is another balancing act as CPU utilisation is directly tied to this. If your iSCSI server is mainly serving workload with a small IO size and requires "snappyness" in response times the better option is to try and use a high interrupt rate. Obviously your CPU will be more busy so if the server has other application requirements it may be best to leave this as the default for general workload setting.

The interrupt settings can be adjusted per queue with the `ethtool -C` command parameter. Be very careful with this as mistakes or incorrect configuration will often lead to a detrimental outcome. From a troubleshooting perspective this may, or better "will" result in elongated diagnostic times and hence delay a solution.

The above settings are mostly fairly common in current network cards. Some advanced features that could be relatively rare or only availble on high end cards are thing like iWarp/RDMA or Data Direct IO.

Check with your vendors administration guides and release notes what each of these mean and what they do. Some of them may use different terminology or do not provide the more advanced features you require. As with many things you basically get what you pay for.

Stats, counters, values The `ethtool -S <NIC>` shows the output of various counters. In troubleshooting scenarios you're most often after the error counters. The values presented are summarised in totals as well as per queue in both rx as tx direction.

```
[admin@server ~]$ ethtool -S eno1
NIC statistics:
     rx_packets: 2700659
```

```
     tx_packets: 27272
     rx_bytes: 346241480
     tx_bytes: 3078465
     rx_pkts_nic: 2700688
     tx_pkts_nic: 27272
     rx_bytes_nic: 357047545
     tx_bytes_nic: 3194663
<snip>
```

A quick look at errors using `grep` is quickly done.

```
[admin@server ~]$ ethtool -S eno1 | grep -i -E "drop|error|eno|fail"
     rx_errors: 0
     tx_errors: 0
     rx_dropped: 63202
     tx_dropped: 0
     rx_over_errors: 0
     rx_crc_errors: 0
     rx_frame_errors: 0
     rx_fifo_errors: 0
     rx_missed_errors: 0
     tx_aborted_errors: 0
     tx_carrier_errors: 0
     tx_fifo_errors: 0
     tx_heartbeat_errors: 0
     rx_length_errors: 0
     rx_long_length_errors: 0
     rx_short_length_errors: 0
     rx_csum_offload_errors: 2
     alloc_rx_page_failed: 0
     alloc_rx_buff_failed: 0
```

Just looking at these values do not provide you with much usefull detail. It is merely a snapshot of the current state but does not show the increments over time. It is therefore not a good indicator to diagnose an issue that has happened yesterday. If a storage problem is currently ongoing you would need to collect the output on a continuous basis and either differentiate them or put them into a graph. From there you can identify when these errors occur and correlate them to specific issues.

If you make use of some advanced features like Intels `ntuples` you can assing specific workloads (in the sense of IP source and TCP port) to specific queues which each can be pinned to a specific interupts and therefore CPU/core. These counters will then be specific to that/those queue(s).

When it comes to the Intel 10G networks cards one of the best writups I've seen is from Joe

Damato over at PackageCloud in collaboration with some awesome folks at Private Internet Access who published a very detailed set of pages on the Linux network stack and how to make correct decisions on which properties. See the references here [40] [41] [42] [43] [44] [45]

Another juggernaut in the networking space is Broadcom. They have a vast array of NICs spanning from 1Gb/s to 200 Gb/s. Talk about speed. phew... Unfortunatly I did not have one of their adapters so could not provide further insight at this stage.

Regarding the advanced features that some NICs provide I would not use them from the get-go. Mainly because you need very tight control on all the settings and as soon as they are set they don't really adapt well to changing conditions from an OS or application perspective. If you run into a different workload profile the configuration you've spend days or weeks to figure out may turn out to become a major bottleneck.

Dealing with the hardware-side of the fence is one of the most complex issues and may very often not result in the outcome required when external factors change. A change in network behaviour may be detrimental to the performance and stability of the iSCSI setup. If a good configuration is found you should also be aware that these are often aligned to the workload that the application throws at it. If the application changes the tuning excersize of the IO and networking stack may need to be reviewed.

10.4.2 TCP/IP

iSCSI runs on TCP simply because it needs a reliable transport mechanism. Depending on the capabilies of the iSCSI server you may need to adjust network traffic profiles on each of the iSCSI initiators.
The assumption is that the iSCSI server is a dedicated system and does not serve another purpose. You then have to determine what the capabilities of that system are relating to cpu, memory but most important of all obtain a baseline on IO possibilities. Test that local server in various IO scenarios as when multiple initiators are connected to this iSCSI server the chances of a mixed workload are very high.

The other part is network throughput. If you have a network interface capable of doing 40G line-rate but your backend disk configuration is not able to sustain that, you will see that, depending on the rate of incomming data with a heavy write workload, a lot of packet drop will occur. This is not only detrimental for performance but also puts the stability of the system at risk and may cause data corruption. There is no way on the server side where you can change or adjust that. Lowering the line-rate of the network interface or adding an ingress queuing discipline on the server side will not resolve that simply because the amount of incoming data cannot be controlled. The only way to do that is to take the baseline numbers from the IO tests you have done, verify if the network stack is capable of handling that and adjust the iSCSI initators accordingly so that the sum of the imposed workload by those initiators does not exceed the capabilities of the server.
With a primarily read based workload the mode is obviously reversed and you may need to

setup network queing disciplines on the server on a per initiator basis.
Whenever you run into IO issues in an iSCSI environment the main cause is often related to the design of the network in relation to the requested or offered workload. In storage environments this becomes even more troubleshome as the IO stack is basically expecting a reliable communications path between the initiator and targets. Although TCP itself is a reliable protocol it does not guarantee an error-free transport. Its smarts are very much in the detection and subsequent actions such as re-transmissions and slow-start (`AIMD`) algorithms. This does not align well with channel protocols and thus a well thought out design and configuration from the start is paramount to such an environment.

Another part of configuration is related to the way TCP is configured on the respective interfaces. Things like TCP windows sizes, selective acknowledgement, congestion control etc all have an influence on the performance and behaviour of traffic.

I mentioned the `ethtool` command before where you can have a look at the counters of the ethernet interface and the `ip -s addr` or `ip -s link` gives you the counters on a the IP layer of the network stack.

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group \
  default qlen 1000
    link/ether 08:00:27:f6:fe:d1 brd ff:ff:ff:ff:ff:ff
    inet 10.10.11.3/24 brd 10.10.11.255 scope global noprefixroute enp0s3
       valid_lft forever preferred_lft forever
    inet6 fe80::e64a:ce43:a75b:27a/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
    RX: bytes  packets  errors  dropped missed  mcast
    9440       49       0       0       0       43
    TX: bytes  packets  errors  dropped carrier collsns
    16540      167      0       0       0       0
```

Any value other than `0` on the `errors`, `dropped` and `missed` column should be further investigated. If they keep increasing over a short period of time you will see IO errors occuring if the upper layer iSCSI stack is not able to correct the IO by retransmitting the request in time.

Tools like `bmon` give you a more dynamic overview of this. If you want to monitor for a longer period of time you may want to use tools like Zabbix or Nagios which also allows for network profiling and therefor can help out with adjusting configuration settings on a iSCSI level as well. I will not go into details on this as this is way outside the scope of the book.

10.5 Fibre Channel

Although technically a transport protocol, I want to give Fibre Channel a dedicated section as it has been the main transport layer of storage infrastructures for over two decades and

still is. It is the choice for large scale storage deployments in most, if not all, Fortune 500 companies for mission critical workloads and performance requirements. I'm going to spend some emphasis on this topic mainly because I've seen, over the course of my career, there is a lot of misconception on how the various bits of the FC protocol work and how they interact with the operating system.

Its first iteration was a so called Arbitrated Loop implementation. If you ever heard of Token Ring in a network topology then you're not far off. It allowed for a shared bus to be created in a loop topology. The AL implementaion is obsolete an no longer implemented in products.

Fibre Channel (FC hereafter) has a history of almost 4 decades and from its inception was designed with storage in mind. This means that it catered to be a very high speed flat network where all traffic flow decisions are made in hardware. As with most stacks, FC is build on various layers which each operate independant but link the layer above and below it.

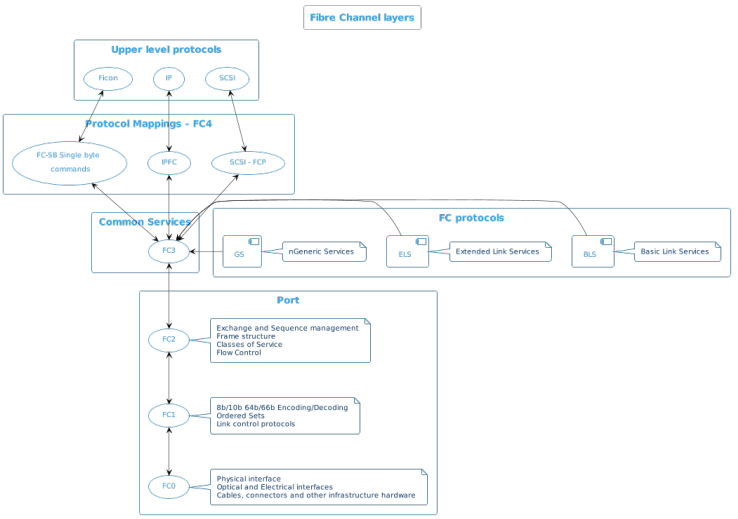


Figure 10.7: Fibre Channel layers

FC is agnostic of channel based protocols which means that the behaviour of the transport layer is adapting to the behaviour of the channel. As an example in the way SCSI works is that it operates asychronously:

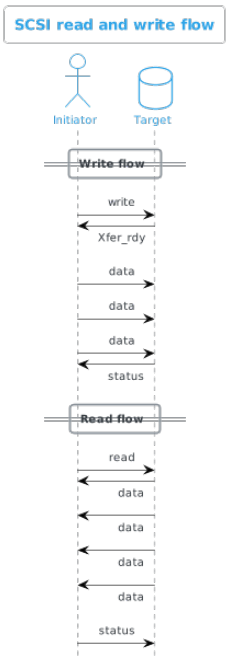


Figure 10.8: SCSI Read and Write Flow

The way FC communicates is aligned to these flow characteristics.

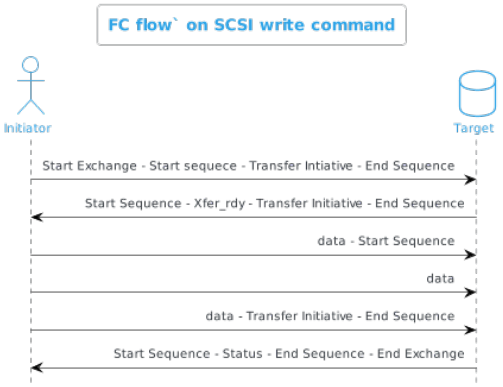


Figure 10.9: FC Flow on SCSI

The above also tells you that a SCSI IO is roughly mapped into a, so called, `exchange`, each change in direction of flow starts a new `sequence` by informing the remote side that the initiative of the flow is transferred to that remote side. Therefore a new sequence gets started. Any subseqent frame going from the same originator will keep the same sequence id. Each sequence has one or more frames either containing commands, data or status messages.

10.5.1 Flow Control

With TCP operating on a sliding window based on a negotiated TCP window sizse for flow control, FC uses buffer to buffer credits. The way this works is that upon link initialisation the two ports on either side of the physical link give each other a fixed amount of credits basically saying "You can send me this amount of frames before you need to halt and wait for me to replenish your used credits. The replenishement of these credits is done with so called `R_RDY` signals.

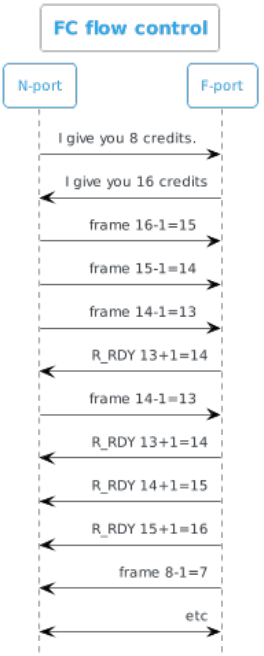


Figure 10.10: FC flow control

The way this works is shown in figure fig. 10.10 on a very high level.

Obviously when the F-port sends frames to the N-port the counters and R_DY primitives

run the opposite way. This behaviour ensures that no frames will be sent from one side to the other without having the assurance the remote side is not able to receive the data. This will result in a reliable dataflow where the sender is ensured that any frame it sends will be delivered.

If life was that good I would not be having a job. :-)

There are a few potential issues with buffer to buffer flow conrtol. The first is when something happens with the R_RDY primitive signal. If that gets lost or corrupted in any way, the side which sent the frame is then not able to replenish the amount of credits. You will then run into scenarios where at some stage the sending side wil not have any credits left and therefore is not allowed to send any frames. That will obviously result in a stalled condition which may have other consequenses as this state will then propagate back into the fabric.

Below is such a representation of a fabric where one link between a host and a switch is observing issues and is no longer able to recieve frames.

10.5.2 Fabrics

Figure 10.11 shows a high level representation of a small two single switch fabrics.

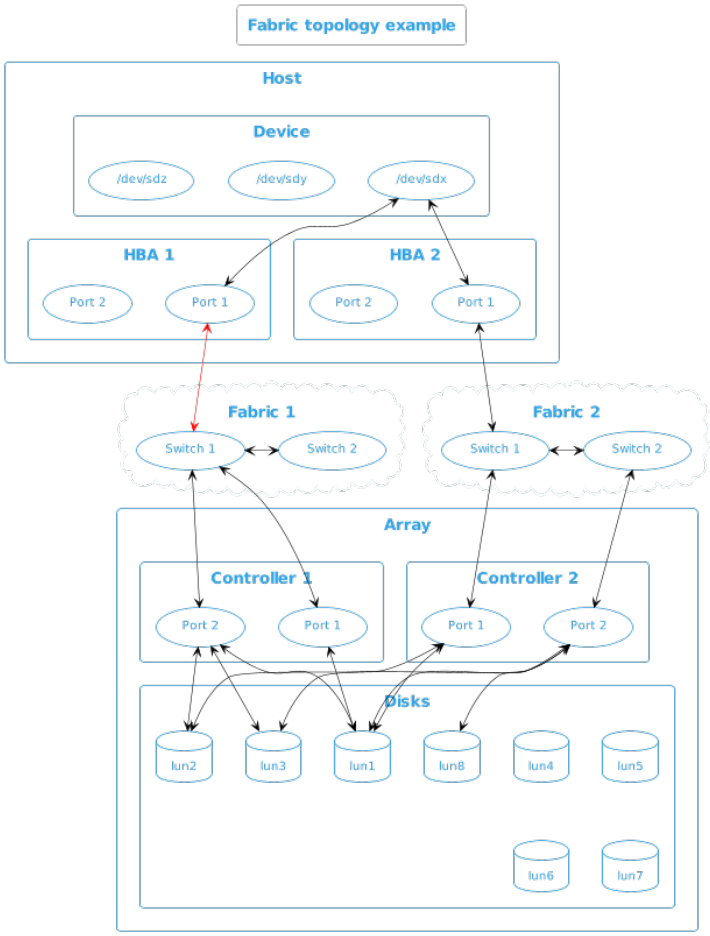


Figure 10.11: Fabric Topology example

Lets assume that controller 1 port 1 needs to send frames to the host. As the HBA port of that host is no longer able to receive those frames the buffers of the switch will fill up and the credit shortage of the switch port will then propagate back to the array port preventing it from sending any frames. Mind the any term here. The above image shows only one host but you can imagine that when more hosts are mapped to the same array port these will also be impacted as the flow control methodology is on a per link basis.

There are a fair few mechanisms implemented over the years which allow for recovery of such situations. The two that will first show up is the Link Reset (LR) protcol and the second

is the BB Credit Recovery mechanism. The LR is the simplest one where on side of the link that is stuck with 0 credits for over a pre-defined period of time will send a link reset primitive notification basically telling the remote side to forget all frames and actual credits and start again with the values that where negotiated during link initialisation. The BB Credit Recovery mechanism uses a counter based table between frames sent and R_RDY received. Some modulo calculations whill then ensure that discrepancies can be corrected. More recent developments have resulted in the so called `FPIN` [46] (Fabric Performance Impact Notification) addition in the FC standards. This allows end devices to become more aware of issues in the fabric and adjust their workload profile and path selection accordingly. At the time of this writing most HBA and switch vendors support this already but many storage vendors have yet to implent this functionality. From a Linux side there are developments ongoing to have one or more parts of the FPIN signals recognised and automatically change IO workflow and path selection based on those changed characteristics. [47]

10.5.3 HBAs

One of the most important pieces in relation to the level of neglect is the HBA. The HBA is almost always overlooked when OS maintenance is planned and executed. It is often seen as a dumb piece of hardware which requires no attention. The opposite is actually the case. The level of firmware and driver enhancements and bugfixes is actually astounding. The configurable options that you actually need to look at is significant. The firmware is mainly responsible for the hardware internals as well as interaction between the card, system hardware and on/offloading of data and FC commands onto the wire. The driver is responsible for interaction with the OS stack as well as the command control between the OS and the external storage systems. Outdated firmware and drivers are a major issue and given the fact that the majority of them are badly maintained they are very often a cause of problems. This is not only true for Linux but is tied to all operating systems which allow for loosly coupled software.

The HBA information exposes itself mainly via locations in `sysfs`.

- /sys/class/fc_host
- /sys/class/scsi_host
- /sys/class/nvme* (In case NVMe over Fabrics is used)

The first one has mainly information on the low level fibrechannel information like the FCID, fabric name, WWN, supported speeds etc. whereas the scsi_host directory provides more info on how the OS can interact with the HBA.

I mentioned earlier in this chapter that errors on a physical layer will cause havoc somewhere in the fabric or network and may have serious consequenses in any part of the fabric if these issues have a severe impact on flow control. On the TCP/IP side you would normally look on the ethernet and TCP side with the `ethtool` and/or any of the IP bases diagnostics tools.

On the FC HBA side you need to look at the files located in the statistics directory of the

/sys/class/fc_host/<hostid>/ location.

```
[root@centos8 storage]# ls /sys/class/fc_host/host11/statistics/
dumped_frames         fcp_frame_alloc_failures  fcp_packet_alloc_failures
link_failure_count    reset_statistics          error_frames
fcp_input_megabytes   fc_seq_not_found          lip_count
rx_frames             fc_no_free_exch           fcp_input_requests
fc_xid_busy           loss_of_signal_count      rx_words
fc_no_free_exch_xid   fcp_output_megabytes      fc_xid_not_found
loss_of_sync_count    seconds_since_last_reset  fc_non_bls_resp
fcp_output_requests   invalid_crc_count         nos_count
tx_frames             fcp_control_requests      fcp_packet_aborts
invalid_tx_word_count prim_seq_protocol_err_count  tx_words
[root@centos8 storage]#
```

Three files in here are particularly interesting.

- reset_statistics
- seconds_since_last_reset
- error_frames

Al the stats you see here are snapshots in time and show a static value representing the state of that counter at that point in time. This means that unless the `seconds_since_last_reset` is not recent (as in minutes or a few hours) you will not be able to say if these errors occured two minutes ago, yesterday, last week or last month. Most counters are also stored in a 32-bit register in the hardware which means that if the counter reaches 4294967296 the register will flip back to 0 just for that counter and you have no way of correlating these individual values which makes troubleshooting efforts on this layer useless.

The first thing you would need to do is create a new baseline on the host as well as the switch where this host is connected to by issuing the command that would reset those as follows.

```
echo 1 > /sys/class/fc_host/*/statistics/reset_statistics
```

On the switch you would either do a `statsclear` (for a Brocade switch) or a `clear counters interface all` (on a Cisco).

Ensure that all counters that represent a physical link issue are checked and that the cause of them are fixed. Be aware that errors in frames, like crc, do not neccesarily indicate a physical link issue as that error might already have been inflickted somewhere upstream.

So now and then you may run into an issue which cannot be explained properly by just looking at the standard events that show up in "/var/log/messages".

Issues such as

```
Oct 7 18:24:20 centos8 kernel: lpfc 0000:81:00.0: 0:1305 Link Down Event \
  xc received Data: xc x20 x800110 x0 x0
Oct 7 18:24:24 centos8 kernel: rport-11:0-4: blocked FC remote port time out: \
  removing target and saving binding
Oct 7 18:24:24 centos8 kernel: lpfc 0000:81:00.0: 0:(0):0203 Devloss timeout \
  on WWPN 50:04:0e:60:07:a3:70:00 NPort x01ee40 Data: x0 x8 x2
```

are fairly common and the above simply shows a Link Down event. These are the most easy to troubleshoot when your remote switchlog tells you:

```
18:26:59.565715 SCN Port Offline;rsn=0x10004,g=0x12 A2,P0 A2,P0 93 NA
18:26:59.565721 *Removing all nodes from port A2,P0 A2,P0 93 NA
18:28:07.998318 SCN LR_PORT(0);g=0x12 A2,P0 A2,P0 93 NA
18:28:08.006029 SCN Port Online; g=0x12,isolated=0 A2,P0 A2,P1 93 NA
18:28:08.007307 Port Elp engaged A2,P1 A2,P0 93 NA
18:28:08.007331 *Removing all nodes from port A2,P0 A2,P0 93 NA
18:28:08.007594 SCN Port F_PORT A2,P1 A2,P0 93 NA
18:28:08.099107 SCN LR_PORT(0);g=0x12 LR_IN A2,P0 A2,P0 93 NA
18:28:20.669283 SCN Port Offline;rsn=0x10004,g=0x14 A2,P0 A2,P0 93 NA
18:28:20.669288 *Removing all nodes from port A2,P0 A2,P0 93 NA
```

as a result of

```
Wed Oct 7 18:28:07 2020 admin, FID 128, 10.10.10.10, portenable 4/29
Wed Oct 7 18:28:20 2020 admin, FID 128, 10.10.10.10, portdisable 4/29
```

Diagnostics becomes more problematic when it is just the events that show the links bounce but show no further information. Obtaining extended information from the HBA drivers may then be very helpful.

10.5.3.1 Update Drivers and Firmware

As you know I'm very picky when it comes to maintenance. If I see cases where System and/or Storage administrators have basically been slacking for a long time the chances are very high that I will tell you that and commence diagnosing issues as soon as these things are all up to date. You wouldn't believe the sheer amount of issues that have been resolved in firmware and drivers over any given time-period.

That being said going to the Linux side of the Emulex (or Broadcom) drivers for the LP31000/LP32000 cards which are very popular in many form-factors.

The driver will show as an lpfc module and is by default compiled into a ramfs image when installed. This will allow the card to be used in a boot-from-san variation if needed. The module will load as such and register with the scsi-subsystem

```
root@centos[~]$ lsmod
<snip>
```

```
lpfc 978944 81
nvmet_fc 32768 1 lpfc
nvme_fc 45056 1 lpfc
scsi_transport_fc 69632 1 lpfc
<snip>
```

With the more recent versions of the driver it will also provide an NVMe_oF initiator and target so that NVM equipment can be utilized when attached to a FC fabric.

10.5.3.2 Log Verbosity

Loggin with an Emulex card can be done on the driver level as well as the HBA firmware. Unless you get some instructions to do so leave the firmware logging as is. Mainly because changing these parameters will require a reload of the driver that basically instructs the firmware logging facility to capture data in some host memory region. Obviously that will involve some engineering efforts to diagnose anyway so that will not be very helpful to yourself or your OEM support-organisation unless it needs escalating to Broadcom/Emulex.

Changing the logging verbosity of the driver itself is much easier but may also incur some performance impact so don't just flick on the "0xFFFFFFFF debug" button. The driver logging facility is a bitmap value based on the below table:

Table 10.3: Emulex logging configuration options

LOG Message	Verbose Mask Definition	Verbose Bit Verbose Description
LOG_ELS	0x00000001	ELS events
LOG_DISCOVERY	0x00000002	Link discovery events
LOG_MBOX	0x00000004	Mailbox events
LOG_INIT	0x00000008	Initialization events
LOG_LINK_EVENT	0x00000010	Link events
LOG_IP	0x00000020	IP traffic history
LOG_FCP	0x00000040	FCP traffic history
LOG_NODE	0x00000080	Node table events
LOG_TEMP	0x00000100	Temperature sensor events
LOG_BG	0x00000200	BlockGuard events
LOG_MISC	0x00000400	Miscellaneous events
LOG_SLI	0x00000800	SLI events
LOG_FCP_ERROR	0x00001000	Log errors, not underruns
LOG_LIBDFC	0x00002000	Libdfc events
LOG_VPORT	0x00004000	NPIV events
LOG_SECURITY	0x00008000	Security events
LOG_EVENT	0x00010000	CT,TEMP,DUMP, logging
LOG_FIP	0x00020000	FIP events

LOG Message	Verbose Mask Definition	Verbose Bit Verbose Description
LOG_FCP_UNDER	0x00040000	FCP underruns errors
LOG_SCSI_CMD	0x00080000	ALL SCSI commands
LOG_NVME	0x00100000	NVME general events
LOG_NVME_DISC	0x00200000	NVME discovery/connect events
LOG_NVME_ABTS	0x00400000	NVME ABTS events
LOG_NVME_IOERR	0x00800000	NVME I/O Error events
LOG_EDIF	0x01000000	External DIF events
LOG_AUTH	0x02000000	Authentication events

 If you don't know what these mean, or have no clue on how to interpret the output, it's not much use changing these values and you should be asking guidance from your support provider. The output will only confuse you and if you don't know what the commands and responses should be, it's only a bunch of hex values.

The values as displayed above can be summed depending on which verbose logging needs to be enabled. For instance if your OEM asks you for Link events, ELS and Initialiation events you may get asked to enable verbose logging with either the "hbacmd" or via "sysfs". The value of the parameter will than be "0x19"

The above are for Broadcom/Emulex adapters only and the options of the logging functionality are tied to the driver and hardware capabilities. These therefor may evolve over time and as such may not be 100% accurate at the time of publishing. I've incorporated them so you know the option is there and you can utilise this functionality. The admin manuals that accompany the drivers will have the latest instructions.

hbacmd or sysfs If you have hbacmd installed any change done in the logging preferences also automatically kicks of dracut and builds a new boot image. The command has a few additional parameters

```
hbacmd setdriverparam 10:00:00:90:fa:c7:cd:f9 G P log-verbose 0x135661
```

The first three a fairly obvious. Command setting driver parameters for PWWN 10:xxxxxx. The G stands for Global basically meaning it is valid for all adapters and the P stands for Permant. That ensures the parameter that follows is also applied after reboots. The log-verbose parameter is basically the configuration what we're adjusting. The 0x135661 is a combination of values obtained via the table above.

The value can also dynamically be applied via sysfs in the "/sys/class/scsi_host/host<X>" (where <X> is the adapter ID) directory. The LPFC driver will create the system file as appropriate in that folder and one of which is indeed the "lpfc_log_verbose" file. The 0x<123456> value can be echoed to that file and the driver will dynamically pick this up.

```
[root@centos8 host11]# cat lpfc_log_verbose
0x0
[root@centos8 host11]# echo 0x135661 > lpfc_log_verbose
```

The change is immediate logged

```
Oct 8 17:03:15 centos8 kernel: lpfc 0000:81:00.0: 0:(0):3053 lpfc_log_verbose \
  changed from 0 (x0) to 1267297 (x135661)
```

When you change all of them with

```
[root@centos8 scsi_host]# echo 0x135661 > host11/lpfc_log_verbose
[root@centos8 scsi_host]# echo 0x135661 > host12/lpfc_log_verbose
[root@centos8 scsi_host]# echo 0x135661 > host13/lpfc_log_verbose
[root@centos8 scsi_host]# echo 0x135661 > host14/lpfc_log_verbose
```

The messagelog will show something similar like this

```
Oct 8 17:28:28 centos8 kernel: lpfc 0000:81:00.0: 0:(0):3053 lpfc_log_verbose \
  changed from -1 (xffffffff) to 1267297 (x135661)
Oct 8 17:28:50 centos8 kernel: lpfc 0000:81:00.1: 1:(0):3053 lpfc_log_verbose \
  changed from 1267297 (x135661) to 1267297 (x135661)
Oct 8 17:28:58 centos8 kernel: lpfc 0000:83:00.0: 2:(0):3053 lpfc_log_verbose \
  changed from 1267297 (x135661) to 1267297 (x135661)
Oct 8 17:29:04 centos8 kernel: lpfc 0000:83:00.1: 3:(0):3053 lpfc_log_verbose \
  changed from 1267297 (x135661) to 1267297 (x135661)
```

The interesting past is that the PCI system paths to the adapter entries are used here. This is reflected in the “0000:81:00.0:”, 0000:83:00:0:” etc entries.

Remember that in normal circumstances you would not need to change these values. The basics are logged anyway and only in specific circumstances you would need to adjust that. Also be aware that using a debug value of 0xFFFFFFFF can incur a significant performance overhead on busy systems as a lot needs to be logged.

Another thing that I get often queried about is which HBA port belongs to which SCSI number.

Identifcation of the respective HBA’s can be done by looking at the adapter entries in the eventlog as mentioned above. In this case the 81 and 83 values are a reflection of the PCI id and the 00.0 and 00.1 are the individual ports on those adapters.

```
81:00.0 Fibre Channel: Emulex Corporation LPe31000/LPe32000 ...
81:00.1 Fibre Channel: Emulex Corporation LPe31000/LPe32000 ...
83:00.0 Fibre Channel: Emulex Corporation LPe31000/LPe32000 ...
83:00.1 Fibre Channel: Emulex Corporation LPe31000/LPe32000 ...
```

You can see these entries coming back in the /sys/class/fc_host directory where logical links to the PCI devices are created

```
lrwxrwxrwx. 1 root root 0 Sep 4 15:08 host11 -> ../../devices/pci0000:80 \
  /0000:80:03.0/0000:81:00.0/host11/fc_host/host11
lrwxrwxrwx. 1 root root 0 Sep 4 15:08 host12 -> ../../devices/pci0000:80 \
  /0000:80:03.0/0000:81:00.1/host12/fc_host/host12
lrwxrwxrwx. 1 root root 0 Sep 4 15:08 host13 -> ../../devices/pci0000:80 \
  /0000:80:03.2/0000:83:00.0/host13/fc_host/host13
lrwxrwxrwx. 1 root root 0 Sep 4 15:08 host14 -> ../../devices/pci0000:80\
  /0000:80:03.2/0000:83:00.1/host14/fc_host/host14
```

As soon as you know this you can associate the respective WWN of the adapter to the one you see on the switch:

```
[root@centos8 ~]# cat /sys/class/fc_host/host11/port_name
0x10000090fac7cde8

switch1:FID128:admin> switchshow
switchName: switch1
switchType: 165.0
<snip>

Index Slot Port Address Media Speed State Proto
==========================================================
66 4 2 01ef40 id N8 Online FC F-Port 50:04:0d:20:10:17:b5:b8
<snip>
93 4 29 010000 id 16G Online FC F-Port 10:00:00:90:fa:c7:cd:e8
```

The above shows you when you see errors happening as part of a SAN attached disk where to look and how to associate the Emulex adapters to the respective WWN's on your SAN.

From there on you can also identify which disks are presented to that adapter. As you've seen above the PCI subsystem creates a host interface per FC port. In my case these are host11 to host14.

A simple way to check is to just do an "ls" on /sys/class/scsi_disk/device/block tree.

```
[root@centos8 scsi_disk]# ls */device/block/
<snip>

'11:0:0:0/device/block/':
sdm

<snip>
```

```
'11:0:0:8/device/block/':
sdu
```

As you can see the 11:xxxxx entries will list the respetive "/dev/sd*" entries that is being used for mounting volumes, MPIO listings etc.

10.5.3.3 Emulex/Broadcom

Emulex was one of the earliest vendors of HBA's and are now part of Broadcom. Together with Qlogic they are one of the mainstream vendors in this area. Emulex delivers the drivers and firmware in two packages. The firmware is loaded statically which means you have to upgrade this manually. The versions are linked to eachother and allthough they allow for some minor version discrepancy it is always adviseable to have the firmware and driver versions aligned. This prevents some weird behaviour where the driver may expect a certain state or result of a system call which the firmware is not able to deliver. Other issues are mainly related to bugs and how the HBA's interact with the physical side of the system as well as datahandling between various parts of the stack it controls like DMA requests etc. The moral of this is mainly ***KEEP YOUR FIRMWARE AND DRIVERS UP TO DATE !!!***.

As the SCSI subsystem and the FC stack have each their own timing settings it is beneficial to pay attention to how these interact and depend on eachother.

The way the HBA interacts with the OS is mainly configured via the module parameters that can either be changed dynamically or statically which, in some occasions, require a host reboot. Consult the documentation for the most up-to-date options and parameters.

The below driver parameters are extremely important as these tie in directly with how various actions in the device mapper multipath daemon are triggered. More on that in the MPIO chapter.

lpfc_nodev_tmo parm: lpfc_nodev_tmo:Seconds driver will hold I/O waiting for a device to come back (int) (This one is mutually exclusive with the lpfs_devloss_tmo, deprecated in later versions)

lpfc_devloss_tmo parm: lpfc_devloss_tmo:Seconds driver will hold I/O waiting for a device to come back (int)

lpfc_poll parm: lpfc_poll:FCP ring polling mode control: 0 - none, 1 - poll with interrupts enabled 3 - poll and disable FCP ring interrupts (int)

lpfc_poll_tmo parm: lpfc_poll_tmo:Milliseconds driver will wait between polling FCP ring (uint)

lpfc_task_mgmt_tmo parm: lpfc_task_mgmt_tmo:Maximum time to wait for task management commands to complete (uint)

10.5.4 Port up sequence

To understand what happens when an HBA logs into a fabric you need to know the order of events and the expected communications between the HBA, switch and array.

Below a simplified overview.

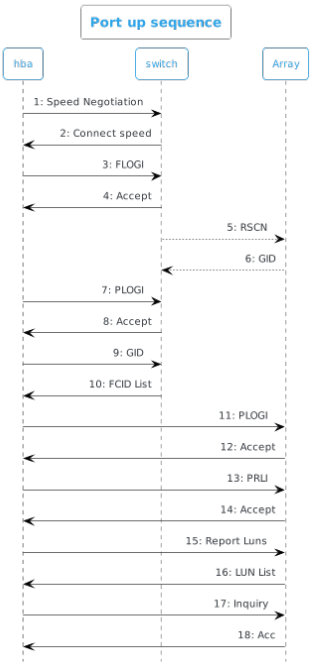


Figure 10.12: Port Up Sequence

Be aware that behind the scenes in the fabris a lot of other communication is taking place to inform other switches of various changes etc.

Let's go thru the steps.

1. After the HBA turns the laser on it will need to determine the speed on which it can communicate
2. The highest speed is negotiated and the state is set

3. The HBA sends a `FLOGI` (Fabric Login) to announce itself to the fabric, obtain a `FCID` (Fabric address) from the F-port controller and sends the service parameters which it supports
4. The F-port controller will send the FCID back and will either acknowldge the service parameters or make some adjustments based on what it supports
5. The switch will send an `RSCN` (Registered State Change Notification) to the array to inform that port something has changed in the fabric. (It is then up to the array if it acts upon that)
6. The HBA will send a `GID` (Get ID) to the fabric nameserver to obtain a list of `FCID`'s it can talk to
7. Followed by a `PLOGI` (Port Login) to the fabric name server along with some service parameters
8. The nameserver will send an `ACK` back to confirm these service parameter
9. The HBA will also send a `GID` to the nameserver obtain a list of N-ports it can talk to (if properly zoned)
10. The nameserver will send back the list of zonemembers to which the HBA is allowed to talk to.
11. The HBA then logs in into the array FC port to exchange detail on the FC side
12. Which the array either accepts or denies. (if an array port is not configured to allow a loging frome that particular HBA the login will be denied)
13. If the login is accepted there needs to be a session on the `FCP` (Fibre Channel SCSI/NVMe) layer established which is done with a `PRLI` (Process Login)
14. The array will send an `ACK` back so that the protocol parameters on that layer is agreed
15. The HBA driver and the Linux IO stack have now created the base configuration for the target side and a Report LUNS is sent to the array
16. The array will then send a list of LUNS back to the host
17. Each lun will then recieve an `Inquiry` from the host in order to retrieve things like size, vendor, model, type and supported options
18. The array will send that information and the host IO stack can then register these devices so they can be used

Now you would say, based on the above, that this is rather simple. Bear in mind though that this representation is an extremely simplified view and that many other commands and parameters go back and forth. You can imagine that when a shared-nothing cluster is connected to the same port using the same LUN's, the array would need to know that and be configured in such a way that access registrations and de-registrations (via so called reserves) are needed so that one host does not overwrite data from another host. There are many more knobs and sliders in that infrastructure that can cause an issue.

10.5.5 Switches

Speaking of FibreChannel there is really no way to ignore the switch-side in order to troubleshoot link related issues. I've written hundreds of articles on my blog over the years to

inform storage administrators many of the nuts and bolts of FibreChannel environments. You can find them over at my blog [48].

If hosts are connected to a FC switch the first thing to look at is to see if there are any physical issues on a link. There is no way to circumvent a physical link problem on any other layer. It would be a bit like trying to use a different set of tyres on a sportscat when the road goes from a smooth bitumen surface to a dirt track riddled with pot holes.

Physical link issues are most often caused by one of the following factors:

- Connectors or patchcords not properly seated
- Connectors are dirty. (Even the slightest specle of dust or debree can cause light signal distortion)
- Cables are broken. (This may cause a dB loss resulting in the amount of light on the receiver side to drop below the receiver sensitivity threshold)
- Incorrect cabling. Mix of various OM type [49] cables causing signal degradation or loss due to refraction and dispersion issues
- SFP's are broken. Either the TX laser is no longer able to send light or has degraded so that there is not enough light seen on the remote site
- SFP's are not supported for the length and speed of the connection. You need to refer to the guidelines of the manufacturer what speeds are supported on which cable and SFP type. Very often the length of the cable leans against the boundaries of what is possible. That leads to signal issues causing link instability and frame corruption.

10.5.5.1 Link Errors

As you've seen on the HBA side the error counters can be easily retrieved via the `/sysfs` interface. You have to be aware though that link errors are unidirectional. This means that counter values you see in these files are errors that have been identified on the receiving side (ie these are incomming errors). It doesn't say that the remote side is free of issues which may still impact the bi-directional traffic sequences.

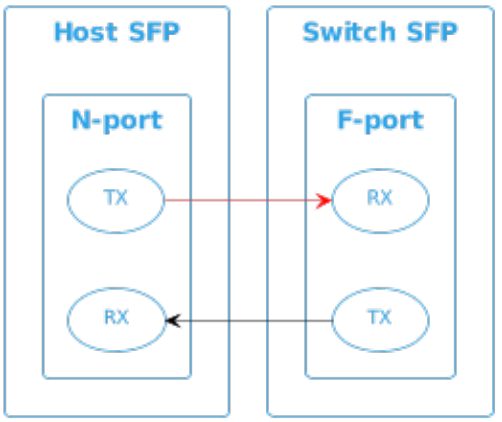


Figure 10.13: SFP Directional Errors

Figure 10.13 shows the physical diagram of the actual link. The arrows show the direction of the data. The red line depicts an issue either on the connector, cable, patch-panel etc. The errors will therefore show up on the switch side and not on any of the counters of the HBA.

This does not mean the host will not observer errors but these are then a conseqeunce of the physical link issue. You may for example see that the switch at some stage will reset a link because it ran out of buffer credits. That will obviously be seen on the host side.

So in order to see if any errors are seen on that switchport you would need to check there. As an example:

```
portshow 64
portDisableReason: None
<snip>
Interrupts:        25          Link_failure: 2          Frjt:         0
Unknown:           0           Loss_of_sync: 0          Fbsy:         0
Lli:               25          Loss_of_sig:  2
Proc_rqrd:         109         Protocol_err: 0
Timed_out:         0           Invalid_word: 0
Tx_unavail:        0           Invalid_crc:  0
Delim_err:         0           Address_err:  0
Lr_in:             3           Ols_in:       2
Lr_out:            2           Ols_out:      3
```

```
porterrshow:
          frames      enc    crc    crc    too    too    bad    enc   disc   \
          link   loss   loss   frjt   fbsy  c3timeout    pcs    uncor
       tx     rx      in    err    g_eof  shrt   long   eof     out   c3    \
        fail    sync   sig                  tx    rx     err    err
 64:    2.5g   2.2g   0      0      0      0      0      0      0     22    \
  2      0      2      0      0      0      0      0      0
```

For link state counters this is the most useful command in the switch however there is a perception that this command provides a "silver" bullet to solve port and link issues but that is not the case. Basically it provides a snapshot of the content of the `LESB` [13] of a port at that particular point in time. It does not tell us when these counters have accumulated and over which time frame. So in order to create a sensible picture of the statuses of the ports we need a baseline. This baseline can be created to reset all counters and start from zero. To do this issue the "statsclear" command on the cli.

There are 7 columns you should pay attention to from a physical perspective.

1. enc_in – Encoding errors inside frames. These are errors that happen on the FC1 with encoding 8 to 10 bits and back or, with 10G or higher, from 64 bits to 66 and back. Since these happen on the bits that are part of a data/command frame, these are counted in this column.

2. crc_err – An enc_in error might lead to a CRC error however this column shows frames that have been market as invalid frames because of this crc-error earlier in the datapath. According to FC specifications it is up to the implementation of the programmer if he wants to discard the frame right away or mark it as invalid and send it to the destination anyway. There are pro's and con's on both scenarios. So basically if you see crc_err in this column it means the port has received a frame with an incorrect crc but this occurred further upstream.

3. crc_g_eof – This column is the same as crc_err however the incoming frames are NOT marked as invalid. If you see these most often the enc_in counter increases as well but not necessarily. If the enc_in and/or enc_out column increases as well there is a physical link issue which could be resolved by cleaning connectors, replacing a cable or (in rare cases) replacing the SFP and/or HBA. If the enc_in and enc_out columns do NOT increase there is an issue between the SERDES chip and the SFP which causes the CRC to mismatch the frame. This could be most likely a firmware issue that controls the interaction between those two. Check with your vendor if this is the case and if updated firmware fixes this problem.

4. enc_out – Similar to enc_in this is the same encoding error however this error was outside normal frame boundaries i.e. no host IO frame was impacted. This may seem

[13] Link Error Status Block

harmless however be aware that a lot of primitive signals and sequences travel in between normal data frame which are paramount for fibre-channel operations. Especially primitives which regulate credit flow. (R_RDY and VC_RDY) and signal clock synchronization are important. If this column increases on any port you'll likely run into performance problems sooner or later or you will see a problem with link stability and sync-errors (see below).

5. Link_Fail – This means a port has received a NOS (Not Operational) primitive from the remote side and it needs to change the port operational state to LF1 (Link Fail 1) after which the recovery sequence needs to commence. (See the FC-FS standards specification for that)

6. Loss_Sync – Loss of synchronization. The transmitter and receiver side of the link maintain a clock synchronization based on primitive signals which start with a certain bit pattern (K28.5). If the receiver is not able to sync its baud-rate to the rate where it can distinguish between these primitives it will lose sync and hence it cannot determine when a data frame starts.

7. Loss_Sig – Loss of Signal. This column shows a drop of light i.e. no light (or insufficient RX power) is observed for over 100ms after which the port will go into a non-active state. This counter increases often when the link-loss budget is overdrawn. If, for instance, a TX side sends out light with -4db and the receiver lower sensitivity threshold is -12 db. If the quality of the cable deteriorates the signal to a value lower than that threshold, you will see the port bounce very often and this counter increases. Another culprit is often unclean connectors, patch-panels and badly made fibre splices. These ports should be shut down immediately and the cabling plant be checked. Replacing cables and/or bypassing patch-panels is often a quick way to find out where the problem is.

8. The other columns are more related to protocol issues and/or performance problems which could be the result of a physical problem but not an underlying cause. In short look at these 7 columns mentioned above and check if no port increases a value.

When you start troubleshooting troubleshooting issues you see on the host, and it has been determined that the cause is most likely external, you need to check in with your storage administrators and have them verify the data path between the server and the host. As any physical problem in the infrastructure is relatively easy to find and would need to be resolved first before proceeding on any other avenue, the link errors as I mentioned before should be checked on each switch that is in that data path. The actual process may get more complicated when the issue remains after any physical issue is resolved but that is really for the storage admins to find and fix.

I've received quite a few question what this CRC and CRC_G_EOF actually is. Normally when a FC port creates a framestructure it has a SOF (Start of Frame), a frame header, the data, a CRC checksum and an EOF (End of Frame). That EoF is tagged with a few

possible options. When the frame shows no problems it is tagged with a `n` so in a trace it would show up a EoFN. As switches do not use a store and forward method of frames but use cut-trough switching, meaning parts of the frame may have already been forwarded to any next potentional hop, there is no way for a port to retrieve or call back the frame when it has determined something is wrong with that frame. The only way it can inform the next port in the data patch is by invalidating the frame by changing the `EoFN` tag to a `EoFi` where the `i` stands for "Invalid". If this happens anywhere in the fabric it is easy to trace this back as the counters in the switches will tell you where the frame was observed of having a CRC error but was not invalidated yet.

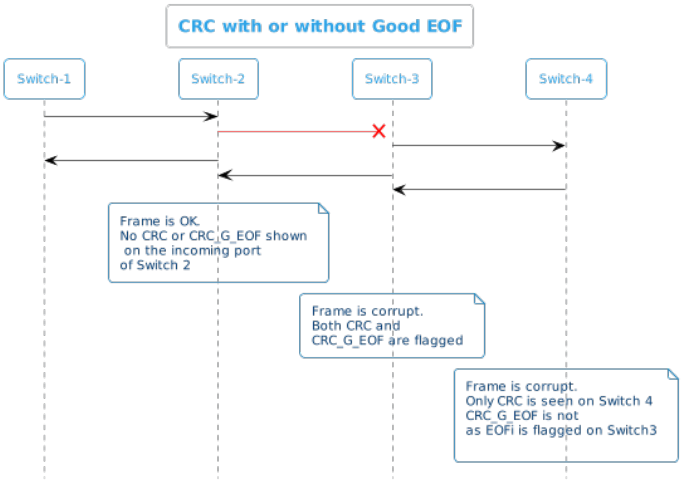


Figure 10.14: CRC Errors

As shown in fig. 10.14 the moment a frame gets corrupted between switch 2 and 3. THe counters on the ingress port of switch 2 will show no increments, the counter on the ingress port of switch 3 will show a `CRC` error as well as a `CRC_G_EOF` as the frame has not been invalidated yet. The ingress port of Switch 3 will change the `EoF` tag from `EoFN` to `EoFI` thereby marking the frame as being invalid. Switch 4 will show only a CRC error and will not increment the `CRC_G_EOF` counter as is it not having a good `N` tag.

If any issues are seen on a physival level you should start correcting them ASAP. There is simply no shortcut or workaround if these physical issues occur. The way to do that is as follows:

1. Start by cleaning the connectors and reseating cables if cable testing equipment is not available. This often clears up minor bit errors.

2. If patch-panels are used, re-route cables to other ports on the panels to see if that resolves the issue.
3. Connect to a different port on the switch, clear port statistics, and monitor the porterrshow output to see if the problem follows the move.
4. Replace cable(s).
5. Replace SFP on the sending side, e.g. this will be either an HBA port or a port on a storage channel host adapter.
6. Replace SFP on the switch side if the problem does not follow the move.

Do them in the order as above as that has proven to be the most effective way.

Even though the above information is obtained from Broadcom hardware, a similar kind of information can be obtained from a Cisco switch albeit the terminology is somewhat different.

Chapter 11

MPIO - MultiPath IO

In environments where IO requests are being sent over a networked transport protocol you most often seen that disks presented out of a storage array are accessed via multiple paths. We've seen this in the FC topology example in fig. 10.11 as well as the iSCSI network fig. 10.5.

As I wrote in one of my articles [2] MPIO is not a IO failsafe mechanism. This means that in case of IO errors, this layer will not be retransmitting IO's when frames are dropped or other, protocol based, errors may occur like SCSI status messages being corrupted or simply time out. That is the responsibility of the transport protocol (SCSI, NMMe etc..).

The MPIO layer is designed to be a path manager where it creates virtual devices when it detects if two or more disks are actually the same. There are basically two methods it can use for this. One is the WWID and the second a manually created alias based on that WwID.

Example:

```
[root@server ~] multipath -ll
mpathr (360060e8007c370000030c37000000705) dm-17 HITACHI,OPEN-V
size=10G features='0' hwhandler='0' wp=rw
`-+- policy='service-time 0' prio=1 status=active
  |- 13:0:3:5 sdah 66:16  active ready running
  `- 14:0:0:5 sdag 66:0   active ready running
```

In the above example the devices "`sdah`" and "`sdag`" are actually the same device presented via FibreChannel out of the same array over different paths. The MPIO software aggregates these in the "`mpathr`" device which can be addressed in the same way as a normal `sdx` disk.

MPIO also monitors the state of accessibility over the paths to each of the physical disks. Depending on the configuration it can do a read on a sector, a TUR (Test Unit Ready) to

that LUN or via a some other ways. The state can be seen in the last three output fields of the physical device "`active ready running`" or via "`multipathd list paths`".

```
[root@server queue]~ multipathd list paths
hcil     dev  dev_t  pri dm_st  chk_st dev_st  next_check
13:0:3:0 sdw  65:96  1   active ready  running XXXXXX.... 13/20
14:0:0:0 sdx  65:112 1   active ready  running XXXXXX.... 12/20
14:0:0:2 sdaa 65:160 1   active ready  running XXXXXXX... 14/20
14:0:0:1 sdy  65:128 1   active ready  running XXXXXXX... 15/20
```

11.1 ALUA

ALUA stands for `A-synchronous Logical Unit Access`. *Ok, what does that mean??* you'd say. As were in the MPIO chapter you can imaging that access to `disks` presented out of storage arrays is one of the most important parameters. Without access you're pretty much lost. Depending on the infrastructure and storage array capabilities you'll most likely have more than one access path to the particular volume located in that array.

When access characteristics are 100% the same over all the paths to that disk you will not see a need for ALUA. When referring to fig. 10.11 you can see that the array consists of two separate controllers. Each controller will serve a volume exclusively and present this out of one or more of its front-end ports. For redundancy purposes it will also present the volume out of the other controller. This is known as an Active/Passive setup. During the discovery phase the host will detect the two access paths and determine this is actually the same volume. It now needs to determine which access path it actually needs to use. The reason is that the controller actively servicing the volume should support the full SCSI command set (as far as the supported functioned and features allow) to service it. The access path on the passive controller still needs to support the command and task management functions however the internal array service capabilities may not be optimal and could incur a performance penalty.

By looking at the standard inquiry response it will see that the volume has a few bits set in the `TPGS` [1] field.

These can be as follows:

[1] Target Port Groups

Table 11.1: TPGS response field.

Code	Description
00b	The logical unit does not support asymmetric logical unit access or supports a form of asymmetric access that is vendor specific. Neither the REPORT TARGET GROUPS command nor the SET TARGET PORT GROUPS command is supported.
01b	The logical unit supports only implicit asymmetric logical unit access. The logical unit is capable of changing target port asymmetric access states without a SET TARGET PORT GROUPS command. The REPORT TARGET PORT GROUPS command is supported and the SET TARGET PORT GROUPS command is not supported.
10b	The logical unit supports only explicit asymmetric logical unit access. The logical unit only changes target port asymmetric access states as requested with the SET TARGET PORT GROUPS command. Both the REPORT TARGET PORT GROUPS command and the SET TARGET PORT GROUPS command are supported.
11b	The logical unit supports both explicit and implicit asymmetric logical unit access. Both the REPORT TARGET PORT GROUPS command and the SET TARGET PORT GROUPS commands are supported.

Based on this it can make an inquiry around the way which path should be used by requesting the remote controller via the `REPORT TARGET PORT GROUPS` command. Based on how the array is configured it can receive one or more TPG descriptor lists each containing the characteristics of that TPG for that associated volume. A descriptor showing which volume presented out of which ports have a preferred access and thus the host can use that information to use the most optimal path to the volume. Unless some special configuration is in place you will most often see either the `active/optimized` or the `active/non-optimized` being returned. As I mentioned before this does depend on the target configuration.

If this field is set to any non-0 value the initiator requests the array for the access charac-

teristics via the REPORT TARGET PORT GROUP command which should return any of the following states

- active/optimized
- active/non-optimized
- standby
- unavailable
- logical block dependent

It goes a bit too far to explain these in detail in this book as the SCSI Primary Commands standards (See the T10 website [50]) has a very detailed chapter specifically dedicated to this functionality.

In the exampled below you see that the volume that is presented out of a iSCSI target has two paths with the same access characteristics and are therefore place into the same policy group even though the priority handler is set to `1 alua`.

```
sclient:~ # multipath -ll
36001405705f3340f4124399b75f0ef3d dm-0 LIO-ORG,idiskc
size=8.0G features='1 queue_if_no_path' hwhandler='1 alua' wp=rw
`-+- policy='service-time 0' prio=50 status=active
  |- 3:0:0:0 sdb 8:16 active ready running
  `- 4:0:0:0 sdc 8:32 active ready running
```

In short `ALUA` allows for a host to array discovery and management mechanism to try and use best path. The definition of `the best path` is however up to interpretation as that also depends on how the rest of the array is configured as well as the state of the infrastructure on that path. It is up to the systems administrator, in cooperation with the storage administrator, to design and configure the best possible setup.

11.2 NVMe

As of around mid-2022 a few patches have been submitted to exclude some vendors NVMe storage arrays from the the device-mapper multipath daemon. The reason is that the NVMe code in the Linux kernel supports native MPIO functionality. That functionality was already part of the specification since version 1.1 dating back to 2011. It is enabled by default in the kernel but can be turned off via the `nvme-core` module parameter.

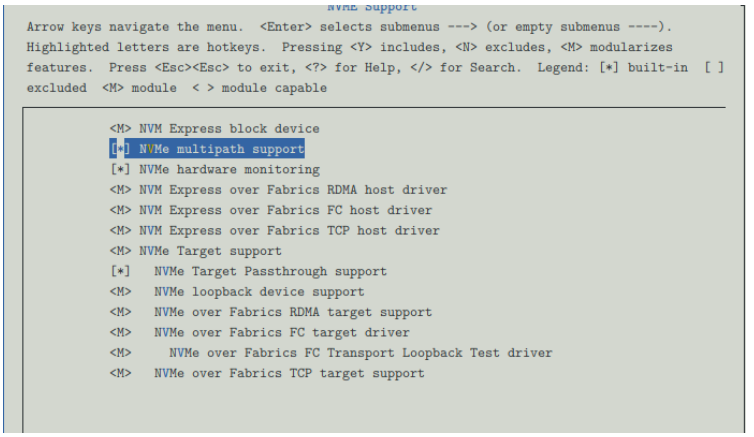


Figure 11.1: NVMe kernel multipath support

The only two options that pertain MPIO for NVMe devices are

```
[root@server 5.17.9-200.fc35.x86_64]# modinfo nvme-core
filename:       /lib/modules/5.17.9-200.fc35.x86_64/kernel/drivers\
  /nvme/host/nvme-core.ko.xz
version:        1.0
license:        GPL
rhelversion:    9.99
<snip>
parm:           multipath:turn on native support for multiple \
  controllers per subsystem (bool)
parm:           iopolicy:Default multipath I/O policy; 'numa' (default)\
  or 'round-robin'
```

I do have some doubts over using the native NVMe multipathing option over the device mapper one. This is mainly because when the transport layer is either `tcp` or `fibre-channel`, the behaviour in error conditions does not have the extended functionality that the `DM-Multipath` solution has. Especially when it comes to intermittent packet or frame drop or losing and regaining connections to remote ports, this may become seriously cumbersome. When taking into account that each NVMe namespace can contain up to 64000 commands, the host and application may have some serious recovery work cut out for them.

What `ALUA` does for SCSI devices, `ANA` support is the NVMe equivalent. It stands for `A-synchronous Namespace Access`. Even though the implementation is somewhat different than ALUA, the methodology is basically the same. A set of inquiries and responses determine the support and access paths to the volumes which in turn determine the multipathing software to configure the respective policies.

Be aware that the terminology and naming conventions change when native multipathing is enabled on NVMe devices and references to controllers and subsystems show the same values however they have a different meaning.

For example the below is an output of a `nvme-cli list` command where native NVMe multipathing is not enabled. The representation of the `/dev/nvmeXnY` output has the `X` as the controller and the `Y` as the namespace.

```
Node SN Model Namespace Usage Format FW Rev
/dev/nvme0n1 0000000000000000 Linux 1 160.82 GB / 160.82 GB 512 B + 0 B 4.4.131-
/dev/nvme1n1 0000000000000000 Linux 1 160.82 GB / 160.82 GB 512 B + 0 B 4.4.131-
```

When however native multipathing is enabled you'll see that the output only shows

```
Node SN Model Namespace Usage Format FW Rev
/dev/nvme0n1 0000000000000000 Linux 1 160.82 GB / 160.82 GB 512 B + 0 B 4.4.131-
```

The `X` no longer represents the controller but now links to the NVMe subsystem. The native multipath logic has aggregated the paths from each of the controllers to that subsystem and namespace. Something to be aware off. The Linux DM multipath driver does have support for native NVMe in the sense that it can query and present the configuration information of the subsystem and namespaces. It does not have the ability to actively interact with the native NVMe multipath module.

11.3 Multipath.conf overrides

When devices presented out of different arrays or from different vendors, they may require one or more settings to be adjusted. Many vendors already contribute to the multipath device-mapper source code and set some default generic settings which should allow for a relatively easy detection, configuration and operation. There are however many occasions where you do need to adjust settings based on application behaviour. This most often has a dependency on timing and IO failure tolerance of the application. That is something that needs to be tested before taking an application into production.

The settings are configured in the `/etc/multipath.conf` file and additional files can be included when placed in the `/etc/multipath/conf.d` directory.

An example is shown below. I've highlighted a few settings that are especially important when configuring the timers for path failures. These settings are primarily for Hitachi based arrays but may be applicable to others as well.

```
[root@server ~]# cat /etc/multipath/conf.d/adjustments.conf

defaults {
 polling_interval 5
 max_polling_interval 15
```

```
 path_checker tur
 failback manual
 marginal_pathgroups yes
 fast_io_fail_tmo 3
 dev_loss_tmo 3600
 detect_checker no
 detect_prio yes
 prio path_latency
 prio_args "io_num=50 base_num=5"
 verbosity 2
}

devices {
 device {
  vendor "(HITACHI|HP)"
  product "^OPEN-"
  path_grouping_policy "multibus"
  path_selector "service-time 0"
  no_path_retry fail

# One  of  the  four  parameters of supporting path check based on accounting IO error
# such as intermittent error. When a # path failed event occurs twice in
# marginal_path_double_failed_time seconds due to an IO error and all the
# other three parameters are set, multipathd will fail the path and enqueue
# this path into a queue of which members are sent a couple # of continuous
# direct reading asynchronous IOs at a fixed sample rate of 10HZ to start IO
# error accounting process.
marginal_path_double_failed_time 10

# One  of  the four parameters of supporting path check based on accounting IO error
# such as intermittent error. If it is set to a value no less than 120, when a
# path fail event occurs twice in marginal_path_double_failed_time second due to
# an  IO  error,  multipathd  will fail the path and enqueue this path into a
# queue of which members are sent a couple of continuous direct reading
# asynchronous IOs at a fixed sample rate of 10HZ to start the IO accounting
# process for the path will last for marginal_path_err_sample_time. If the
# rate of IO error on a particular path is greater than the
# marginal_path_err_rate_threshold, then the path will not reinstate for
# marginal_path_err_recheck_gap_time  seconds  un-less there is only one active
# path. After marginal_path_err_recheck_gap_time expires, the path will be requeued
# for rechecking. If checking result is good enough, the path will be reinstated.
marginal_path_err_sample_time 30
```

```
# The error rate threshold as a permillage (1/1000). One of the four parameters
# of supporting path  check  based  on  accounting  IO  error  such  as
# intermittent error. Refer to marginal_path_err_sample_time. If the rate
# of IO errors on a particular path is greater than this parameter, then the
# path will not reinstate for marginal_path_err_recheck_gap_time seconds unless
# there is only one active path.
marginal_path_err_rate_threshold 5

# One  of  the four parameters of supporting path check based on accounting
# IO error such as intermittent error. Refer to marginal_path_err_sample_time.
# If this parameter is set to a positive value, the failed path of which the IO
# error rate is larger than marginal_path_err_rate_threshold will be kept in
# failed state for marginal_path_err_recheck_gap_time seconds. When
# marginal_path_err_recheck_gap_time seconds expires, the path will be
# re-qeued  for checking. If checking result is good enough, the path will
# be reinstated, or else it will keep failed.
marginal_path_err_recheck_gap_time 120
 }
}

multipaths {
# As an example a Hitachi GAD Volume uses ALUA as path priority selector.
# This should normally be picked up by the "detect_prio yes" value
# in the defaults section. If that does not work there may be something wrong
# in the array settings. To force the path selector "alua" to be used for
# the dm-entries you know are GAD volumes you can use the below. As there
# are no special identifiers the WWID need to be used. Check with
# "lsblk -o NAME,VENDOR,TYPE,MODEL,WWN,UUID" to get the WWID of the volume
# and use that as "wwid" identifier. Be aware the multipaths section does
# not allow regular expressions!
 multipath {
  wwid 310060e8007c370000030c37000000708
  prio alua
  prio_args alua
 }
}

blacklist {
# Blacklist devices not capable of multipathing.
# This includes most internal disks and disks served by single raid-controllers
 protocol "scsi:unspec"
```

```
 protocol "scsi:ata"
 protocol "undef"
}
```

Be aware that many vendors may use some form or proprietary method for any of the multipath settings. You should always follow that guidance in order nor to be surprised if things may not work as expected. Some areas where this may particularly be of interest is any form of access to some sort of storage clustering mechanism where you may see two luns presented out of two different arrays separated over mid to long distance.

11.4 IO errors

As mentioned the multipath device mapper daemon will only act on event where the transport layer (SCSI in most cases) logs an event that access to a device has disappeared. This is shown in the flowchart in fig. 11.2. When a path is in the active state the multipathd daemon on the host will not interfere or keep track of IO errors. It does however have the ability to periodically check if access to a disk via a path is still there. It does that via the `polling_interval` setting which is then gradually reduced up to the `max_polling_interval` setting. The reason you need to enable this is mainly if paths are used in a non-active-active fashion.
What I mean by that is when a SAN or other network topology is used and devices are set to be accessed in some sort of failover mode, there will not be any IO's issued to any of the passive paths. As errors will only be detected when IO's fail you may run into a situation where a passive path fails but this is not seen by the protocol layer and will therefore not be propagated to the multipath layer. In case an active path fails, for whatever reason, it may not be able to fail over to any of the "surviving" paths as these are now deemed failed as well.

The polling mechanism ensures that all paths are periodically checked via the defined `path_checker` setting.

11.5 Path failure

So what makes a path become unavailable then you might ask? There are a few answers here.

- On a network based infrastructure the iSCSI daemon on the server side has deregistered a disk from a target group and has either notified the clients directly or in-directly (via iSNS) that the disk is no longer presented via that target group.
- A TCP session for an iSCSI connection has closed or is in a FIN_WAIT state.
- On a FC based infrastructure the array has unmapped a disk from one of the port where it was presenting the disk out of. It may either send a `RSCN` [2] directly to the

[2] Registered State Change Notification

nodes that are mapped to that port or may notify the fabric of a change who will then send an RSCN

- A physical link that carried one of the paths either failed or went offline. This will then cause the fabric to send an RSCN resulting in the disk to become unmapped.
- A WWN is deregistered from a fabric nameserver resulting in the device becoming unknown in the fabric.

11.6 Path integrity

Fail back of failed paths should normally be done after a manual verification as, especially with older versions of the kernel and the multipath daemon, there are limited provisions to really check on the stability and integrity of that path. If a disk is presented out of a network or fibre-channel based infrastructure and that is observing intermittent IO errors there is not much that can be done from an MPIO layer or any other part of the host OS except not using that path at all. Automatically failing back to a path that has observed issues is not likely becoming healthy just because a few "test IO's" were successful. Some of these issue are very hard to diagnose especially if the entire path itself does not show any symptoms of physical link issues but are more related to packet/frame timeouts which could be a result of congestion and/or delays. It is still worthwhile to not automatically reinstate a path and try to find the bottleneck in this case.

11.7 Error flow chart

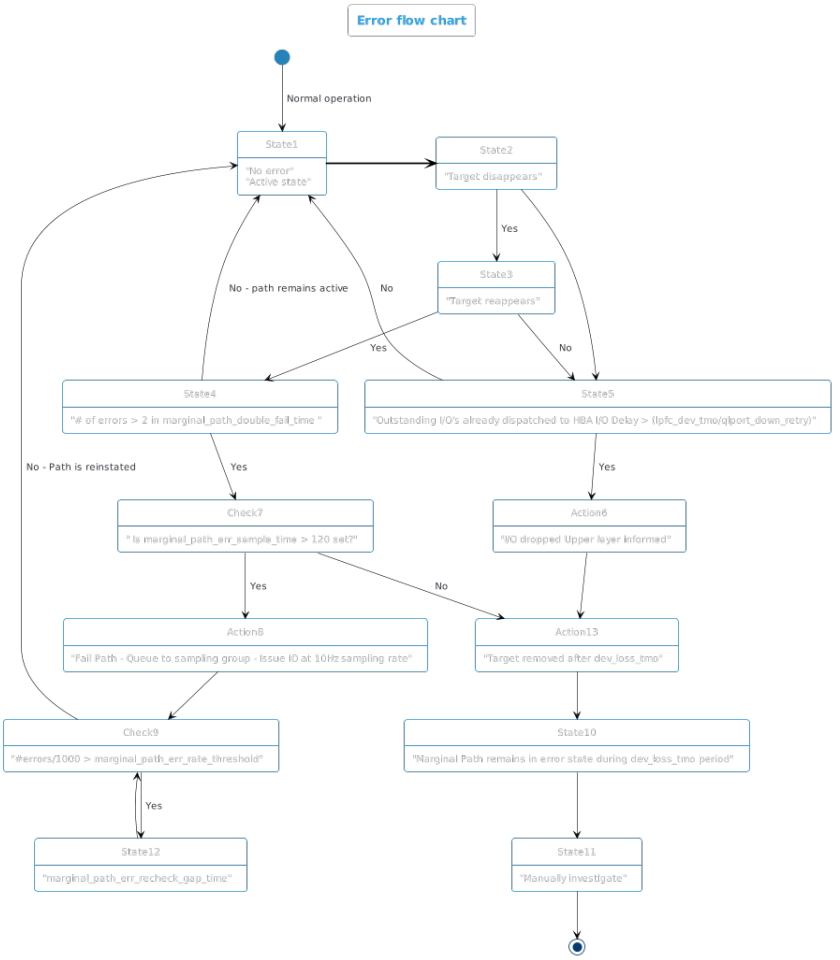


Figure 11.2: MPIO error flow chart

Figure 11.2 shows on a high level how the multipath daemon roughly acts on events. Be aware that this flowchart is depending on the settings in the multipath.conf or any of the configurations set in the kernel.

Chapter 12

Vendor support

In a fair few occasions the troubleshooting efforts you've done may not have resulted in the restoration of the services required and involvement of the support vendor is required. Now, be aware that the support organisation of the respective vendor does not know your environment and does not know your actual problem. Logging a ticket which says `My filesystem does not mount, please let me know how to resolve this` does not really indicate what the problems is and the description is merely a symptom caused by another problem. There is a expectation from any vendor that the system administrator has done a reasonable amount of troubleshooting themselves before lodging a ticket.

12.1 Safeguarding system state

If you've ever watched a CSI [1] series, you know that forensic investigators do not want anyone contaminating the scene of the incident as it may result in findings to become inconclusive. The same is true for systems analysis required from your support vendor. If a problem is observed which you think would need the help of your support vendor, ensure that a support bundle has been created before you start your own troubleshooting efforts. Skipping this may result in log files wrapping, process no longer showing the state they are in when the problem was observed, kernel modules loaded (or not loaded) with the parameters being active, the actual state of any of the device(s) in play etc.

All of the above may change if certain actions are done on a system and most of them will change for sure when a system reboots. In order to prevent this from happening ensure that a system support bundle is created. If the problem cannot be resolved by your own troubleshooting effort and a ticket needs to be opened with your support vendor, you immediately have this information at hand.

[1] Crime Scene Investigation

On RHEL use sos report with or without the various parameters you need (see the –help output) and on SLES use the supportconfig tool (there is also a YAST GUI extension) to collect the system information needed to diagnose the issue in the state when the problem was observed.

```
[root@rhel-84-1 ~]# sos rep

sosreport (version 4.2)

This command will collect diagnostic and configuration information from
this Red Hat Enterprise Linux system and installed applications.

An archive containing the collected information will be generated in
/var/tmp/sos.wc0duw5c and may be provided to a Red Hat support
representative.

Any information provided to Red Hat will be treated in accordance with
the published support policies at:

        Distribution Website : https://www.redhat.com/
        Commercial Support   : https://www.access.redhat.com/

The generated archive may contain data considered sensitive and its
content should be reviewed by the originating organization before being
passed to any third party.

No changes will be made to system configuration.

Press ENTER to continue, or CTRL-C to quit.
<snip>
[plugin:wireless] skipped command 'iwlist scanning': \
  required kmods missing: cfg80211.
 Running plugins. Please wait ...

  Finishing plugins              [Running: subscription_manager]
  Finished running plugins
Creating compressed archive...

Your sosreport has been generated and saved in:
 /var/tmp/sosreport-rhel-84-1-2022-07-07-kwfydmy.tar.xz

 Size 24.82MiB
 Owner root
```

```
 sha256 6c593dff1a6797dd4b41b751f3fd62b0558c234cbeee5c3bd8940735d9cdba9e

Please send this file to your support representative.
```

And the output from SUSE:

```
================================================================================
                     Support Utilities - Supportconfig
                          Script Version: 3.1.11-29.1
                         Library Version: 3.1.11-29.2
                          Script Date: 2022 02 02

 Detailed system information and logs are collected and organized in a
 manner that helps reduce service request resolution times. Private system
 information can be disclosed when using this tool. If this is a concern,
 please prune private data from the log files. Several startup options
 are available to exclude more sensitive information. Supportconfig data is
 used only for diagnostic purposes and is considered confidential information.
 See http://www.suse.com/company/policies/privacy/
================================================================================

Gathering system information
  Data Directory:    /var/log/scc_sserver_220708_1033

  Basic Server Health Check...                 Done
  RPM Database...                              Done
  <snip>
  Local/Warn Logs...                           Done

Creating Tar Ball

==[ DONE ]======================================================================
  Log file tar ball: /var/log/scc_sserver_220708_1033.txz
  Log file size:     2.1M
  Log file md5sum:   c9d655f947a712bb9c1fc5019e531da8-f
================================================================================
```

Keep a detailed track record with timelines and commands issued as well as any other findings you may have encountered during your troubleshooting efforts. This will allow support to build a picture of your environment and will help expedite the path to a resolution.

12.2 Opening tickets

There needs to be a detailed description of the issue

- Description
  - Provide a detailed description.
  - What did you observe.
  - What were the commands you executed when the issue surfaced.
  - Has anything changed on the system itself (patched deployed, selinux policies updated ..... etc...)
  - Has anything changed in the environment? (Network, SAN, Firmware updates.....)
  - Have other systems observe issues as well? (Often when certain external changes are done, it may have an impact on other parts of the infrastructure)
- Time notation
  - When did it start (show detailed timelines and mention the actual time on the system as well as timezone offsets if applicable)
  - It does not help when the time mentions something like `yesterday afternoon` as that may not be reflecting the time the support people look at your case plus the `yesterday` is only valid for one day and your afternoon is likely not their afternoon.
  - Use date and time stamps like `01-Jul-2022 11:45:00 GMT` or, even better, use the timestamp format that your system is using (see the output of the `timedatectl` command)
  - If multiple systems are involved also ensure that, if there are any, time/date differences are shown and these are mentioned when the ticket is opened.
- Place in the infrastructure
  - Ensure that an up-to-date diagram is available which shows the direct connectivity to switches and their upstream/downstream configuration. Outdated diagrams are not useful and will only lead to more confusion and therefore adding to the delay in resolving the problem.
  - Check if addresses (MAC, IP, WWNN/WWPN, FCID's) are correctly shown and align with the actual state of the current configuration of the infrastructure.

The above list is non-exhaustive and also depends on the actual issue at hand. Different vendors may require different information requirements and the level of support is often tied to the entitlements as outlined in the contracts you have with them.

For the latest information and requirements visit the RedHat and SUSE support pages.[51] [52]. Depending on the hardware and overall infrastructure configuration you also may want to involve other teams of your organisation as most often these issues could become complex and a multi-disciplinary approach may be required.

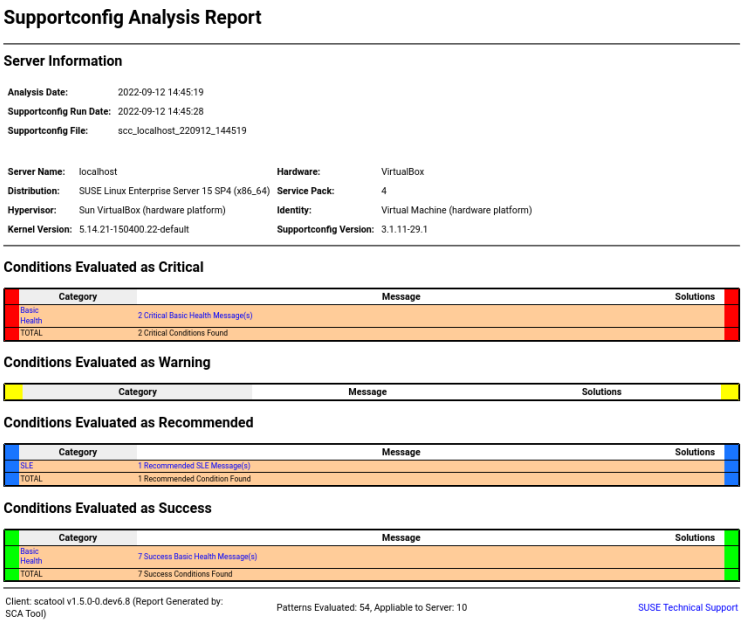
SUSE also has a very handy tool called `SCA` [2] which parses the output of the supportconfig and presents this in HTML format with a detailed level of information around issues seen on the system. Problems that are known by SUSE very often have a knowledge base article attached. This tool can also be set up in your local environment on a dedicated server so `supportconfig` collections can be regularly checked against known issues. The `SCA` tool and the accompanied patterns for the latest SLES releases are regularly updated in the standard SUSE repositories so updating it should be a breeze.

```
root@sserver:~>scatool -s -o .
################################################################################
#   SCA Tool v1.5.1-1
################################################################################

Running Supportconfig On:      sserver
Gathering Supportconfig:       [==================================================]
Processing Directory:          /var/log/scc_sserver_220909_102119/
Total Patterns Available:      0
Pattern Definition Filter:     local
Total Patterns to Apply:       0
Analyzing Supportconfig:       [==================================================]
Applicable Patterns:           0
Pattern Execution Errors:      0
SCA Report File:               ./scc_sserver_220909_102119_report.html

root@sserver:~>
```

[2] Supportconfig Analysis



Supportconfig Analysis Report

Server Information

Analysis Date:	2022-09-12 14:45:19
Supportconfig Run Date:	2022-09-12 14:45:28
Supportconfig File:	scc_localhost_220912_144519

Server Name:	localhost	**Hardware:**	VirtualBox
Distribution:	SUSE Linux Enterprise Server 15 SP4 (x86_64)	**Service Pack:**	4
Hypervisor:	Sun VirtualBox (hardware platform)	**Identity:**	Virtual Machine (hardware platform)
Kernel Version:	5.14.21-150400.22-default	**Supportconfig Version:**	3.1.11-29.1

Conditions Evaluated as Critical

Category	Message	Solutions
Basic Health	2 Critical Basic Health Message(s)	
TOTAL	2 Critical Conditions Found	

Conditions Evaluated as Warning

Category	Message	Solutions

Conditions Evaluated as Recommended

Category	Message	Solutions
SLE	1 Recommended SLE Message(s)	
TOTAL	1 Recommended Condition Found	

Conditions Evaluated as Success

Category	Message	Solutions
Basic Health	7 Success Basic Health Message(s)	
TOTAL	7 Success Conditions Found	

Client: scatool v1.5.0-0.dev6.8 (Report Generated by: SCA Tool) Patterns Evaluated: 54, Appliable to Server: 10 SUSE Technical Support

Figure 12.1: SCA Reporting Tool

Example of the SCA reporting tool

A somewhat different overview can be obtained from RedHat with the difference being that a RHEL system needs to be licensed and under an active maintenance contract. The RedHat portal can then give you an overview of the various packages that are installed and which system health advisories are applicable. It does not provide a troubleshooting analysis in the sense it that it does not diagnose `sosreports`. I'm not aware of any public available option from RedHat that would provide the same/similar functionality that SUSE has with the SCA reporting tool.

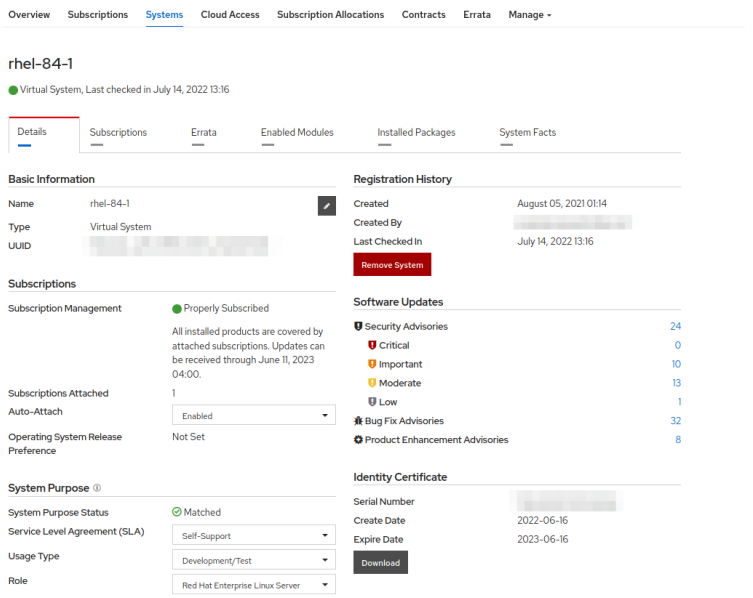


Figure 12.2: Redhat Portal System overview

12.2.1 Severity and Criticality

Make a proper assessment of both and simply be honest with your vendors. If the scale of a issue is large you may flag this as a high severity problem but if this happens in a part of you infrastructure primarily erected for testing purposes the business criticality may not be far reaching. Ensure that if both these characteristics are flagged as business critical and you require the problem to be worked on around the clock by your respective vendor(s), it is expected that personnel of the teams involved from your organisation are also available 24x7, in case information is needed or solution proposals need to be implemented. The severity and criticality works both ways.

12.3 Cross Vendor support

In almost all occasions when it pertains storage related cases you'll see that two or more vendors are involved. Server hardware, host operating system, ethernet and/or fibre-channel switches, storage array vendors and maybe even a few more when other tools or appliances are in use. Many of them are member of `TSANet` [3] [53] which is a non-profit organisation

[3] Technical Support Alliance

that provides vendors with a platform on how to engage each other in case of a mutual customer issue. Please mind the word `mutual` as you as a customer would need to have support contracts in place with every vendor involved. Secondly you would also need to lodge a ticket with each of them and provide these vendors with the ticket numbers so they can cross-reference them. This allows the respective support organisations to discuss the issue at hand and come up with a solution as quickly as possible.

Be aware that TSANet is a vendor organisation and not something you as a customer or interested party can join. It could however show you which of your vendors are a member. Inform you support person that tickets have been opened with these other vendors and they can be contacted via the `TSANet` pathways.

To you the reader

I would like to thank you for purchasing this book and I hope it will help you in your tasks as a Linux administrator dealing with storage related problems. If you would like to see any topic in the book in more detail, please let me know as well. I will do my best to make it as valuable as possible for you.

As I mentioned in the introduction this book is a start to try and aid Linux administrators to troubleshoot storage related issues. As the Linux ecosystem is vast and it's varieties of storage related implementations differs between each distribution this book will need to be regarded as a guideline and not the all-encompassing goto publication for everything related to storage related issues.

The book will be a living thing and I'll try and do my best to keep it up to date and as accurate as possible. The plan is to release an updated version every 9 to 12 months. Any more would not be very useful as the storage industry is in general relatively conservative and new developments take a fair time to make it upstream in the kernel and subsequently in the various distributions.

That being said there may be occasions in the book that may be out of date at a certain time or are incorrect. I've done my best to deep-dive as much as possible in the various topics but there may be occasions some things could have been interpreted incorrectly or have a different meaning resulting in incorrect statements. If you see any irregularities please do not hesitate to contact me. I'm a very approachable fella and would like to help out by keeping this book as good as it gets. Your help will be much appreciated.

Kind regards,
Erwin

http://erwinvanlonden.net  @elonden  https://www.linkedin.com/in/evanlonden

References

[1] Thomas-Krenn.AG, "Home." [Online]. Available: https://www.thomas-krenn.com/en/index.html

[2] E. van Londen, "The great misunderstanding of MPIO," 01-Oct-2012. [Online]. Available: https://erwinvanlonden.net/2012/10/the-great-misunderstanding-of-mpio/

[3] "Oracle Database Documentation." [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/index.html

[4] VanMSFT, "Overview of SQL Server on Linux - SQL Server." [Online]. Available: https://docs.microsoft.com/en-us/sql/linux/sql-server-linux-overview

[5] "Choosing the Right Storage Engine." [Online]. Available: https://mariadb.com/kb/en/choosing-the-right-storage-engine/

[6] "MySQL :: MySQL 8.0 Reference Manual :: 16 Alternative Storage Engines." [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/storage-engines.html

[7] "Overview of the Linux Virtual File System — The Linux Kernel documentation." [Online]. Available: https://www.kernel.org/doc/html/latest/filesystems/vfs.html

[8] "Byte," *Wikipedia*. 12-Apr-2022 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Byte&oldid=1082283626

[9] "International System of Units," *Wikipedia*. 28-Apr-2022 [Online]. Available: https://en.wikipedia.org/w/index.php?title=International_System_of_Units&oldid=1085103501

[10] *NYLUG Presents: Ted Ts'o on the Ext4 Filesystem (Jan 10, 2013) (HD)*. [Online]. Available: https://www.youtube.com/watch?v=2mYDFr5T4tY

[11] "Silicon Graphics," *Wikipedia*. 02-Jan-2022 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Silicon_Graphics&oldid=1063320268

[12] "XFS FAQ - xfs.org." [Online]. Available: https://xfs.org/index.php/XFS_FAQ#Q:_How_to_calculate_the_correct_sunit.2Cswidth_values_for_optimal_performance

[13] C. Dunlop, "Highly reflinked and fragmented considered harmful?" [Online]. Available: https://lore.kernel.org/linux-xfs/20220509024659.GA62606@onthe.net.au/

[14] "Status - btrfs Wiki." [Online]. Available: https://btrfs.wiki.kernel.org/index.php/Status

[15] "Dm-crypt — The Linux Kernel documentation." [Online]. Available: https://www.kernel.org/doc/html/latest/admin-guide/device-mapper/dm-crypt.html

[16] "Linux Unified Key Setup," *Wikipedia*. 16-Jun-2022 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Linux_Unified_Key_Setup&oldid=1093468632

[17] "Home · Wiki · cryptsetup / cryptsetup · GitLab." [Online]. Available: https://gitlab.com/cryptsetup/cryptsetup/-/wikis/home

[18] "GUID Partition Table," *Wikipedia*. 07-Jun-2022 [Online]. Available: https://en.wikipedia.org/w/index.php?title=GUID_Partition_Table&oldid=1091965474

[19] "Roderick W. Smith." [Online]. Available: https://www.rodsbooks.com/

[20] "Endianness," *Wikipedia*. 16-Aug-2022 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Endianness&oldid=1104776037

[21] "Specifications | Unified Extensible Firmware Interface Forum." [Online]. Available: https://uefi.org/specifications

[22] "Cache-policies.rst « device-mapper « admin-guide « Documentation - kernel/git/torvalds/linux.git - Linux kernel source tree." [Online]. Available: https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/Documentation/admin-guide/device-mapper/cache-policies.rst

[23] "The Sg3_utils package." [Online]. Available: http://sg.danny.cz/sg/sg3_utils.html

[24] "BFQ IO scheduler." [Online]. Available: https://www.kernel.org/doc/Documentation/block/bfq-iosched.txt

[25] "Kyber IO scheduler." [Online]. Available: https://www.kernel.org/doc/Documentation/block/kyber-iosched.txt

[26] "Deadline IO scheduler." [Online]. Available: https://www.kernel.org/doc/Documentation/block/deadline-iosched.txt

[27] J. Hands, "Home." [Online]. Available: https://nvmexpress.org/

[28] "libATA Developer's Guide." [Online]. Available: https://www.kernel.org/doc/htmldocs/libata/index.html

[29] "BPF Documentation — The Linux Kernel documentation." [Online]. Available: https://www.kernel.org/doc/html/latest/bpf/index.html

[30] "Ftrace - Function Tracer — The Linux Kernel documentation." [Online]. Available: https://docs.kernel.org/trace/ftrace.html

[31] *BPF Compiler Collection (BCC)*. IO Visor Project, 2022 [Online]. Available: https://github.com/iovisor/bcc

[32] "Sun Microsystems," *Wikipedia*. 22-Apr-2021 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Sun_Microsystems&oldid=1019261568

[33] "Network File System," *Wikipedia*. 17-Mar-2021 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Network_File_System&oldid=1012708858

[34] *Open-iscsi/open-iscsi*. Open-iSCSI, 2021 [Online]. Available: https://github.com/open-iscsi/open-iscsi

[35] "P802.1DC – Quality of Service Provision by Network Systems |." [Online]. Available: https://1.ieee802.org/tsn/802-1dc/

[36] "SCSI Architecture." [Online]. Available: https://www.t10.org/scsi-3.htm

[37] M. Chadalapaka, J. Satran, D. Black, and K. Meth, "Internet Small Computer System Interface (iSCSI) Protocol (Consolidated)." [Online]. Available: https://tools.ietf.org/html/rfc7143

[38] M. Chadalapaka, J. Satran, D. Black, and K. Meth, "Iscsinaming." [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7143#section-4.2.7.4

[39] "Rfc4171." [Online]. Available: https://datatracker.ietf.org/doc/html/rfc4171

[40] J. Damato, "Monitoring and Tuning the Linux Networking Stack: Receiving Data," 22-Jun-2016. [Online]. Available: https://blog.packagecloud.io/eng/2016/06/22/monitoring-tuning-linux-networking-stack-receiving-data/

[41] "Linux networking stack from the ground up, part 1," 22-Jan-2016. [Online]. Available: https://www.privateinternetaccess.com/blog/linux-networking-stack-from-the-ground-up-part-1/

[42] "Linux networking stack from the ground up, part 2," 23-Jan-2016. [Online]. Available: https://www.privateinternetaccess.com/blog/linux-networking-stack-from-the-ground-up-part-2/

[43] "Linux networking stack from the ground up, part 3," 28-Jan-2016. [Online]. Available: https://www.privateinternetaccess.com/blog/linux-networking-stack-from-the-ground-up-part-3/

[44] "Linux networking stack from the ground up, part 4," 30-Jan-2016. [Online]. Available: https://www.privateinternetaccess.com/blog/linux-networking-stack-from-the-ground-up-part-4/

[45] "Linux networking stack from the ground up, part 5," 03-Feb-2016. [Online]. Available: https://www.privateinternetaccess.com/blog/linux-networking-stack-from-the-ground-up-part-4-2/

[46] E. van Londen, "FPIN - The Holy Grail of SAN Stability," 22-Feb-2021. [Online]. Available: https://erwinvanlonden.net/2021/02/fpin-the-holy-grail-of-san-stability/

[47] "[Dm-devel] dm-multipath - IO queue dispatch based on FPIN Congestion/Latency notifications." [Online]. Available: https://www.mail-archive.com/dm-devel%40redhat.com/msg20575.html

[48] "Storage & Beyond." [Online]. Available: https://erwinvanlonden.net/

[49] "Multi-mode optical fiber," *Wikipedia*. 06-Nov-2021 [Online]. Available: https://en.wikipedia.org/w/index.php?title=Multi-mode_optical_fiber&oldid=1053891338

[50] "T10 Technical Committee." [Online]. Available: https://www.t10.org/index.html

[51] "How to engage with Red Hat support." [Online]. Available: https://access.redhat.com/start/how-to-engage-red-hat-support

[52] "Technical Support Guide | SUSE." [Online]. Available: https://www.suse.com/support/handbook/

[53] "TSANet." [Online]. Available: https://tsanet.org/