Сервисы хранения данных

Рассматриваем удаленное хранение файлов (NAS) с использованием протоколов NFS, SMB. А также блочное хранение (SAN) при помощи iSCSI.

Оглавление

SAN и NAS

Storage Area Network (SAN) и Network Attached Storage (NAS) — протоколы, которые предоставляют сервисы хранения данных, при этом, NAS — сервер, предлагающий удалённый доступ к файлам, а SAN — некая отдельная сеть, предполагающая блочный доступ к данным. Хранение блоков, в отличие от файлов, лежит в основе работы обычного жёсткого диска: файлы разбиваются на части одинакового размера, и физически хранятся на устройстве. Блок данных может содержать как файл целиком, так и какую-то его часть. Поверх блочного устройства должна быть установлена файловая система, которая представляет собой логику хранения блоков данных.

В случае с NAS данные хранятся уже поверх файловой системы и представляют собой файлы и папки, доступные удалённым клиентам. То есть NAS-клиент получает уже готовую иерархию с правами, именами файлов и так далее. Загрузить операционную систему с NAS невозможно (исключая вариант с загрузкой по сети PXE boot), в отличие от SAN.

NFS

NFS (network file system) — распределённая файловая система. NFS-сервер позволяет создавать один или несколько дисков, которые могут быть использованы NFS-клиентами. Примонтированные NFS-диски со стороны клиента выглядят так же, как и локальные.

NFS предоставляет клиентам прозрачный доступ к файлам и файловой системе сервера. В отличие, например, от FTP, протокол NFS даёт доступ только к тем частям файла, к которым обратился процесс. Основное достоинство его в том, что он делает этот доступ прозрачным, то есть любое приложение клиента, которое может работать с локальным файлом, также может взаимодействовать и с NFS-файлом, без каких-либо модификаций самой программы.

NFS-клиенты получают доступ к файлам на NFS-сервере путём отправки RPC-запросов на сервер. Это может быть реализовано с использованием обычных пользовательских процессов, а именно, NFS-клиент может быть пользовательским процессом, который осуществляет конкретные RPC-вызовы на сервер, который также может быть пользовательским процессом.

NFS строится по крайней мере из двух основных частей: сервера и одного или большего количества клиентов. Клиент обращается к данным, находящимся на сервере, в режиме удалённого доступа. Для того, чтобы это нормально функционировало, необходимо настроить и запустить несколько процессов. Для этого реализация NFS состоит из нескольких компонентов. Некоторые из них локализованы либо на сервере, либо на клиенте, а некоторые используются на обеих сторонах соединения. Некоторые компоненты не необходимы для обеспечения основных функциональных возможностей, но составляют часть расширенного интерфейса NFS.

Протокол NFS определяет набор запросов (операций), которые могут быть направлены клиентом к серверу, а также набор аргументов и возвращаемые значения для каждого запроса. Первая версия этого протокола существовала только в недрах Sun Microsystems и никогда не была выпущена. Все реализации NFS (в том числе NFSv3) поддерживают вторую версию NFS (NFSv2), которая впервые была выпущена в 1985 году в SunOS 2.0. Третья версия протокола была опубликована в 1993 году.

Протокол удаленного вызова процедур (RPC, Remote Procedure Call) определяет формат всех взаимодействий между клиентом и сервером. Каждый запрос NFS посылается как пакет RPC. На обработке сообщений на сервере работают следующие демоны:

- **rpc.nfsd** — основной демон сервера NFS — nfsd (в новых версиях иногда называется nfsd4). Этот демон обслуживает запросы клиентов NFS. Параметр NFSDCOUNT в файле /etc/sysconfig/nfs в Centos определяет число запускаемых демонов, по умолчанию — 8 (RPC-программа 100003).
- **rpc.mountd** — демон монтирования NFS — mountd. Обрабатывает запросы клиентов на монтирование каталогов. Демон mountd работает на серверах NFS (RPC-программа 100005).
- **rpc.statd** — демон наблюдения за сетевым состоянием (он же Network Status Monitor, он же NSM). Позволяет корректно отменять блокировку файлов после сбоя/перезагрузки. Для уведомления о сбое использует программу /usr/sbin/sm-notify. Демон statd работает как на серверах, так и на клиентах. Ранее он был необходим для работы rpc.lockd, но за блокировки сейчас отвечает ядро (RPC-программа 100021 и 100024 в новых версиях).
- **rpc.lockd** — демон блокировки lockd (он же NFS lock manager (NLM). Обрабатывает запросы на блокировку файлов. Демон блокировки работает как на серверах, так и на клиентах. Клиенты запрашивают блокировку файлов, а серверы её разрешают. Его функции в современных дистрибутивах (с ядром старше 2.2.18) выполняет ядро (lockd) (RPC-программа 100024).
- **rpc.idmapd** — демон idmapd для NFSv4 на сервере преобразует локальные uid/gid пользователей в формат вида имя@домен. Сервис на клиенте преобразует имена пользователей/групп вида имя@домен в локальные идентификаторы пользователя и группы (согласно конфигурационному файлу /etc/idmapd.conf).

Клиент также может запустить демон **nfsiod**. Демон обслуживает запросы, поступающие от сервера NFS. Он необязателен, но увеличивает производительность. В NFSv4 при использовании Kerberos дополнительно запускаются демоны:

- **rpc.gssd** — демон NFSv4, обеспечивает методы аутентификации через GSS-API

  (Kerberos-аутентификация). Работает на клиенте и сервере.
- **rpc.svcgssd** — демон сервера NFSv4, который обеспечивает проверку подлинности клиента на стороне сервера.

Демоны старых версий (NFS v.3 и ниже):

- **nfslogd** — демон журналов NFS, фиксирует активность для экспортированных папок, работает на серверах NFS.
- **rpc.rquotad** — сервер удаленных квот, предоставляет информацию о квотах пользователей в директориях, может работать как на серверах, так и на клиентах.

Кроме указанных выше пакетов, для корректной работы NFSv2 и v3 требуется дополнительный пакет **portmap** (в более новых дистрибутивах заменён на **rpcbind**). Sun RPC — это сервер, который преобразует номера программ RPC (Remote Procedure Call) в номера портов TCP/UDP.

Portmap оперирует несколькими сущностями:

- RPC-вызовами или запросами.
- TCP/UDP портами, в зависимости от версии протокола (TCP или UDP).
- RPC program #.

Демон **portmap** запускается до старта NFS-сервисов.

Работа сервера RPC (Remote Procedure Call) заключается в обработке RPC-вызовов (т. н. RPC-процедур) от локальных и удаленных процессов. Используя RPC-вызовы, сервисы регистрируют или удаляют себя в/из преобразователя портов (portmap, portmapper, он же, в новых версиях, **rpcbind**), а клиенты с помощью RPC-вызовов направляют запросы к portmapper и получают необходимую им информацию.

Следует понимать, что взаимодействие между клиентом и сервером происходит открытым текстом (plain text), следовательно, может быть перехвачено и прочитано злоумышленником. Защитить сам NFS-сервер и взаимодействие сервера с клиентом можно при помощи нескольких методов, которые могут быть использованы вместе:

1. Настройка firewall.
2. TCP Wrappers.
3. SELinux.
4. Kerberos, аутентификация и шифрование.

NFS в CentOS 7

CentOS 7 из коробки поставляется с NFSv4, но при этом поддерживается и NFSv3. Разница между NFSv3 и NFSv4 заключается во взаимодействии между сервером и клиентом, максимальным размером файла и поддержкой WIndows-style для списков контроля доступа (ACLs)
Если вы планируете использовать NFSv4, нет необходимости в установке Remote Procedure Call (RPC) и настройке взаимодействия с **rpcbind**, но это требуется при использовании NFSv3.

Протокол NFSv3 включил в себя поддержку 64-битного размера файла, что позволило обрабатывать файлы размером больше 2 ГБ. Версия 4 расширила эти возможности, а также представила несколько улучшений производительности, улучшенное взаимодействие с Kerberos. Изменился механизм работы блокировки файлов (file locking). Если в версии 3 использовался сторонний NLM (Network Lock Manager), то у NFSv4 нативная поддержка.

NFSv4 поддерживает кластерные решения при помощи pNFS (parallel NFS). Это позволяет масштабировать NFS-сервер горизонтально, добавляя несколько серверов в решение.

Установка NFS

Если мы хотим иметь сервер, который даёт возможность использовать различные протоколы файлового и блочного доступа (SAMBA, CIFS, iSCSI, NFS), можно использовать group install:

```
[root@nfs-server ~]# yum group install "File and Storage Server"
```

Если же необходим только NFS-сервер, наш выбор — nfs-utils

```
[root@nfs-server ~]# yum install nfs-utils -y
================================================================================
 = Package                  Arch         Version                  Repository
                                    Size
================================================================================
= Installing: nfs-utils               x86_64       1:1.3.0-0.61.el7
base       410 k
Installing for dependencies:
gssproxy                x86_64       0.7.0-21.el7             base       109 k
keyutils                x86_64       1.5.8-3.el7              base        54 k
libbasicobjects         x86_64       0.1.1-32.el7             base        26 k
libcollection           x86_64       0.7.0-32.el7             base        42 k
libevent                x86_64       2.0.21-4.el7             base       214 k
libini_config           x86_64       1.3.1-32.el7             base        64 k
libnfsidmap             x86_64       0.25-19.el7              base        50 k
libpath_utils           x86_64       0.2.1-32.el7             base        28 k
libref_array            x86_64       0.1.5-32.el7             base        27 k
libtirpc                x86_64       0.2.4-0.15.el7           base        89 k
libverto-libevent       x86_64       0.2.5-4.el7              base       8.9 k
quota                   x86_64       1:4.01-17.el7            base       179 k
quota-nls               noarch       1:4.01-17.el7            base        90 k
rpcbind                 x86_64       0.2.0-47.el7             base        60 k
tcp_wrappers           x86_64       7.6-77.el7               base        78 k

Transaction Summary
================================================================================
=
```

В зависимости от требований вы можете включить следующие пакеты:

- **nfs4-acl-tools** — предоставляет cli-команды для работы с ACL в NFS-файловых хранилищах (“шарах”).
- **portreserve** — поддержка сервиса portreserve, преемник portmap для NFS-коммуникаций. Предотвращает возможность NFS использовать порты, необходимые другим сервисам.
- **quota** — добавляет поддержку квот для NFS-директорий.
- **rpcbind** — включает поддержку RPC-взаимодействий.

Базовая настройка NFS-сервера

Базовая настройка NFS достаточно проста. Всё, что вам надо сделать — **export** директории на сервере, чтобы примонтировать её на клиенте.

Не забывайте, что соответствующие порты в firewall должны быть открыты, а SELinux — корректно настроен (если используется). NFS контролируется несколькими systemd-юнитами:

- **nfs-secure-server.service** запускает **rpc.svcgssd**-демон, который предоставляет Kerberos-аутентификацию и шифрование для NFS-сервера.
- **nfs-secure.service** запускает **rpc.gssd**-демон, который согласовывает параметры Kerberos-аутентификации и шифрования между NFS-клиентом и сервером.
- **nfs-idmap.service** запускает **rpc.idmapd**-демон, который транслирует user и group IDs в имена. Автоматически запускается nfs-server-юнитом.
- **Nfs-lock.service** необходим для NFSv3. Запускает **rpc.statd**-демон, который предоставляет блокировку, а также статус текущих используемых файлов.
- **nfs-mountd.service** запускает **rpc.mountd**-демон. Необходим для NFSv3.
- **nfs-rquotad.service** запускает **rpc.rquotad**-демон, который предоставляет сервис-квот для NFS-шар. Автоматически запускается nfs-server-юнитом.
- **rpcbind.service** запускает **rpcbind**-демон, который транслирует RPC-номера программ в адреса (порты). Необходим для NFSv3. Автоматически запускается nfs-server-юнитом.

Чтобы запустить NFS-сервер, вам не требуется запоминать все эти юниты. Всё, что нужно — запустить nfs-server-юнит и все зависимости запустятся автоматически.

```
[root@nfs-server ~]# systemctl start nfs-server
[root@nfs-server ~]# systemctl enable nfs-server
Created symlink from
/etc/systemd/system/multi-user.target.wants/nfs-server.service to
/usr/lib/systemd/system/nfs-server.service
```

Проверяем статус сервера:

```
[root@nfs-server ~]# systemctl status nfs-server
● nfs-server.service - NFS server and services
    Loaded: loaded (/usr/lib/systemd/system/nfs-server.service; enabled; vendor
preset: disabled)
   Active: active (exited) since Sun 2019-07-21 11:39:05 CEST; 41s ago
 Main PID: 19150 (code=exited, status=0/SUCCESS)
   CGroup: /system.slice/nfs-server.service

Jul 21 11:39:05 centos7 systemd[1]: Starting NFS server and services...
Jul 21 11:39:05 centos7 systemd[1]: Started NFS server and services.
```

Если необходима поддержка Kerberos, вам следует активировать **nfs-secure-server** и **nfs-secure**-сервисы на сервере и клиенте соответственно.

Управление файлами и доступом

NFS включает в себя широкий набор команд для настройки и последующего управления экспортом (exports), проверки доступных директорий, статистики и так далее. Практически весь набор команд (исключая команды, связанные с mount) можно найти в папке /usr/sbin.

Примонтировать NFS-директорию можно при помощи **mount.nfs**, размонтирование, соответственно, происходит при помощи **umount.nfs**. Так же есть 2 симлинка — **mount.nfs4** и **umount.nfs4**. Функционально они работают так же, как и обычные mount/umount, но, как можно понять из самой команды, относятся к nfs- или nfsv4-системам. Как и любая другая mount-команда, она является эквивалентом варианта mount -t. То есть **mount.nfs4** — то же самое, что и **mount -t nfs4**

Если вы пытаетесь примонтировать шару через **mount.nfs** или **mount -t nfs** без указания версии протокола, сначала NFS-клиент будет использовать nfsv4, и, если сервер ее не поддерживает, откатится к nfsv3.

Наиболее часто используемые команды для конфигурации и тестирования nfs:

- **exportfs** — используется для управления директориями, расшаренными и сконфигурированными через файл /etc/exports.
- **nfsiostat** показывает статистику по I/O-рейту для существующих точек монтирования. Использует информацию из /proc/self/mountstats.
- **nfsstat** показывает статистику по клиенту/серверу, основываясь на текущих точках монтирования. Использует информацию из /proc/self/mountstats.
- **showmount** показывает расшаренные NFS-директории.

Кроме этого, есть набор ACL-связанных команд из пакета nfs4-acl-tools. Вы можете их использовать, если примонтированная шара использует acl-опцию монтирования. Команды достаточно просты и вы можете установить **nfs4_setfacl**, модифицировать **nfs4_editfacl** и показать **nfs4_getfacl** текущий ACL для файла.

Предположим, что вы смонтировали каталог /mnt/nfs/var/nfsshare с опцией acl при помощи протокола NFS. Если вы примените команду **nfs4_getfacl** к файлу или папке в этом общем каталоге, увидите следующий вывод:

```
[root@nfs-client ~]# nfs4_getfacl /mnt/nfs/var/nfsshare

# file: /mnt/nfs/var/nfsshare
A::OWNER@:rwaDxtTcCy
A::GROUP@:rxtcy
A::EVERYONE@:rxtcy
```

ACL установлен, чтобы разрешать (Allow A) или запрещать (Deny D) доступ к файлу или директории владельцу, группе или всем (Owner, Group, Everyone) .
Список прав:

- **r** (read-data (files)) — чтение файлов/директорий.
- **w** (write-data (files)) — создание файлов/директорий.
- **a** (append-data (files)) — модификация файлов.
- **x** (execute (files)) — исполнение файлов.
- **d** (delete the file/directory) — удаление файла или директории.
- **D** (delete-child) — удаление всех вложенных директорий, начиная с текущей.
- **t** (read the attributes of the file/directory) — чтение атрибутов файлов/директорий.
- **T** (write the attribute of the file/directory) — запись атрибутов файлов/директорий.
- **n** (read the named attributes of the file/directory) — чтение назначенных атрибутов.
- **N** (write the named attributes of the file/directory) — запись назначенных атрибутов.
- **c** (read the file/directory ACL) — чтение fACL.
- **C** (write the file/directory ACL) — запись fACL.
- **o** (change ownership of the file/directory) — смена владельца файла/директории.

Модифицировать file ACL можно при помощи **nfs4_setfacl -e filename**. Эта команда откроет текстовый редактор по умолчанию, который позволит вам модифицировать права. Например, запрещаем владельцу писать в файл:

```
[root@nfs-client ~]# nfs4_setfacl -e /mnt/nfs/var/nfsshare/test_file
## Editing NFSv4 ACL for file: /mnt/nfs/var/nfsshare/test_file
D::OWNER@:wa
A::OWNER@:rtTcCy
A::GROUP@:rtcy
A::EVERYONE@:rtcy
```

Настройка NFS-сервера

Настройка директорий на nfs-сервере, доступных для NFS-клиентов, производится в файле **/etc/exports**. После настройки директории могут быть экспортированы при помощи **exportfs -a**. Каждая строчка в файле **/etc/exports** указывает на директорию, которая будет доступна клиентам, а также опции, применимые к ней. Вы можете прописать несколько опций подряд, но саму директорию в файле можно указать лишь один раз. Например:

```
[root@nfs-client ~]# cat /etc/exports
/pub    user1.example.com(rw,sync) *(ro,sync)
/home    *.example.com(rw,async) 192.168.1.0/24(ro)
/tftp    nodisk.example.com(rw,no_root_squash,async)
/var/nfsshare    *(rw,sync,no_root_squash,no_all_squash,acl)
```

В этом примере директория:

- **/pub** экспортирована для пользователя (клиента) user1.example.com, даёт возможность чтения и записи (rw), при этом для всех остальных nfs-клиентов (*) эта директория доступна только для чтения (ro).
- **/home** экспортирована для чтения и записи для для всех клиентов домена example.com (*.example.com), но доступна только для чтения клиентам в подсети 192.168.1.0/24.
- **/tftp** экспортирована для чтения и записи для всех пользователей (даже для рута) для хоста nodisk.example.com.
- **/var/nfsshare** доступна для чтения и записи для всех хостов и любых юзеров на этих хостах.

Опечатки в файле /etc/exports могут привести к неожиданным результатам. Например, пробел в конце строки с директорией для экспорта приведет к синтаксической ошибке и сломает экспорт, а пробел между хостнеймом и скобками с опициями откроет доступ к директории для любых хостов. Пробел между опциями экспорта также недопустим.

Основные опции:

- **async** — операция записи происходит асинхронно. Повышает производительность, но может привести к потере данных при падении nfs-сервера
- **hide** — скрывает структуру файловой системы. Если вы экспортируете директорию, которая включает в себя поддиректории, они также должны быть явно экспортированы.
- **ro** — экскрирует только для чтения.
- **rw** — экспортирует для чтения и записи
- **sync** — записывает данные на диск сервера перед тем как подтвердить успешную запись клиенту
- **all_squash** — транслирует всех локальных и удаленных юзеров в anonymous-юзера.
- **anongid**=*groupid* — указывает groupid для anonymous-юзера.
- **anonuid**=*userid* — указывает userid для anonymous-юзера.
- **insecure** — поддерживает соединения при помощи портов > 1024. В основном, необходимо для NFSv2 и NFSv3.
- **no_root_squash** — обрабатывает root-пользователя NFS-клиента как локального root-пользователя. Без этого параметра root-пользователь будет транслирован в nfsnobody-пользователя
- **sec**=*value* — указывает опции безопасности, например krb5, krb5i, sys и так далее.

Активация экспорта

Сразу после настройки /etc/exports и выполнения **exportfs -a** директории становятся доступными для клиентов. Однако если вы планируете переместить шару или удалить ее, вам следует остановить текущий экспорт. Сделать это можно при помощи **exportfs -ua**. После окончания модификации файла с настройками возобновить экспорт можно, выполнив **exportfs -a** или **exportfs -r.** Разница между **-a** и **-r** незначительна, но важна: **-a** экспортирует все директории, указанные в файле **/etc/exports**, **-r** переэкспортирует директории, синхронизируя список шар и удаляя те, которые отсутствуют в **/etc/exports**.

Доступные для клиента директории можно увидеть при помощи команды **showmount -e *servername***.

Например, в нашем случае — **showmount -e nfs-server.lab**:

```
[root@nfs-client ~]# showmount -e  nfs-server.lab Export list for nfs-
server.lab:
/tftp         *
/home         *
/pub          *
/var/nfsshare *
```

Порты для NFS-взаимодействия

NFSv4 несколько проще для настройки с точки зрения файерволов. Чтобы разрешить соединение между клиентом и сервером, нужно только открыть TCP-порт 2049.

В случае с CentOS 7, этот порт уже является частью nfs-сервиса в **firewalld** (в нашем случае — для зоны **public**):

```
[root@nfs-server   ~]#firewall-cmd   --get-active-
zones public interfaces: ens33

[root@nfs-server ~]#firewall-cmd --permanent --zone=public --add-
service=nfs
[root@nfs-server ~]#firewall-cmd --reload
```

Для NFS3 конфигурация будет немного более комплексной, так как, наряду с самим NFS-сервером, требуются еще вспомогательные сервисы **rpc-bind**, а также **mountd**.

```
[root@nfs-server   ~]#firewall-cmd   --get-active-
zones public interfaces: ens33

[root@nfs-server ~]#firewall-cmd --permanent --zone=public --add-service=nfs
[root@nfs-server ~]#firewall-cmd --permanent --zone=public --add-service=mountd
[root@nfs-server       ~]#firewall-cmd       --permanent       --zone=public
--add-service=rpc-bind
[root@nfs-server ~]#firewall-cmd --reload
```

Как только NFS-сервер был запущен, у вас появляется возможность посмотреть все используемые порты при помощи команды rpcinfo -p.

```
[root@nfs-server ~]# rpcinfo -p
   program vers proto   port  service
   100000    4   tcp    111  portmapper
   100000    3   tcp    111  portmapper
   100000    2   tcp    111  portmapper
   100000    4   udp    111  portmapper
   100000    3   udp    111  portmapper
   100000    2   udp    111  portmapper
   100005    1   udp  20048  mountd
   100005    1   tcp  20048  mountd
```

```
100024    1   udp  50253  status
100005    2   udp  20048  mountd
100005    2   tcp  20048  mountd
100024    1   tcp  56545  status
100005    3   udp  20048  mountd
100005    3   tcp  20048  mountd
100003    3   tcp   2049  nfs
100003    4   tcp   2049  nfs
100227    3   tcp   2049  nfs_acl
100003    3   udp   2049  nfs
100003    4   udp   2049  nfs
100227    3   udp   2049  nfs_acl
100021    1   udp  53130  nlockmgr
100021    3   udp  53130  nlockmgr
100021    4   udp  53130  nlockmgr
100021    1   tcp  44674  nlockmgr
100021    3   tcp  44674  nlockmgr
100021    4   tcp  44674  nlockmgr
```

Исходя из вывода мы можем увидеть номер RPC-программы, версию NFS, протокол, номер порта и сервис, который этим пользуется.

Ограничения NFS

Stateless

NFSv3 — это stateless-протокол (не сохраняет состояние от предыдущих запросов). NFS-клиент взаимодействует с rpc.mountd на сервере. Демон rpc.mountd обрабатывает запрос на монтирование и проверяет, если указанная директория велидна. Если это так, то rpc.mountd предоставляет NFS file handle (aka magic cookie), который в дальнейшем используется между клиентом и сервером для этой шары.

Stateless-взаимодействие позволяет NFS-клиенту ждать ответа от NFS-сервера, даже если тот был, например, перезагружен. В такой ситуации NFS-клиент может зависнуть или даже перезагрузить операционную систему.
Это также может приводить к проблемам блокировки файлов (file lock). Например, файл был открыт одним клиентом, а сервер перезагрузили (что приводит к тому, что блокировка файла не отменяется), и после этого он оказывается недоступным для любых других клиентов.

Эти проблемы привели к тому, что был разработан NFS4, который является уже stateful-протоколом.

Root Squash

Стандартно, NFS выполняет **root_squash**, чтобы не допустить root-пользователей с хостов клиентов иметь root-доступ к NFS-директории на NFS-сервере. Для этого root-пользователь на клиенте (user ID 0) транслирован в nfsnobody — непривилегированного пользователя на сервере.

Эта настройка может быть изменена при помощи опции экспорта **no_root_squash**. Для директорий с **no_root_squash**-опцией, root-пользователь со стороны клиента получает root-доступ к директории на сервере. Несмотря на то, что это может быть достаточно удобным и/или необходимым, вы должны понимать, что это достаточно серьезный риск с точки зрения безопасности.

NFS Hangs

Так как NFSv3 — это stateless-протокол, NFS-клиент может ждать ответа от сервера бесконечно долго. Во время ожидания любой процесс, который захочет посмотреть, записать, создать файлы или директории на nfs-шаре, абсолютно так же зависнет.

Как только это произойдёт, будет достаточно непросто отмонтировать nfs-шару, особенно если вы не использовали lazy-опцию к umount-команде (umount -l). Чтобы избежать таких проблем, вам следует:

- Построить надежную и отказоустойчивую сеть между nfs-клиентом и сервером.
- Монтировать редко используемые nfs-шары только тогда, когда это надо. После использования они должны быть отмонтированы.
- Не используйте async-опцию. Это спасёт вас по меньшей мере от потери данных в случае краша сервера.
- Уберите nfs-директории из пути поиска пользователей, особенно root-пользователя.
- Не монтируйте nfs-шары в корень файловой системы (/). Создайте для них отдельный раздел.

Inverse DNS

Когда NFS-демон проверяет запросы на монтирование, сначала он смотрит на список экспорта в /etc/exports. Далее он берёт IP-адрес клиента, чтобы узнать его хостнейм. Это требует обратной записи в DNS (PTR-запись).

После полученный хостнейм проверяется на разрешенные в файле **/etc/exports**. Если NFS-сервер не сможет найти хостнейм, то rpc.mountd запретит доступ к экспорту для такого запроса.

File locking

Несколько NFS-клиентов могут одновременно примонтировать одну и ту же директорию на сервере. Кроме этого, несколько компьютеров клиентов могут запрашивать доступ к одному и тому же файлу внутри этой директории. Для обработки таких запросов и управления доступом используется file-locking-демон.

NFS традиционно имеет серьезные проблемы с локами. Поэтому, если у вас есть приложение, которая зависит от локов через NFS, вам следует протестировать его работоспособность.

Кроме этого, вам не следует делить одну и ту же директорию между NFS и Samba, так как они используют разные механизмы локов, что может привести к потере данных и повреждению файлов.

Увеличение производительности NFS

- Стандартно NFS запускает 8 процессов для обработки запросов. Если вам кажется, что этого недостаточно, вы можете изменить параметр **RPCNFSDCOUNT** в **/etc/sysconfig/nfs**-файле.
- Если приложение работает медленно, а потеря данных для него не проблема, то вы можете использовать опцию async. Это разрешает серверу отвечать положительно на запрос о записи на диск еще до того, как она была произведена. Но при потере питания либо внезапной перезагрузке сервера такие данные будут утеряны.
- NFS-сервер вынужден производить DNS-лукап достаточно часто. Для ускорения этого процесса и создания кеша вы можете запустить Name Switch Cache Daemon (nscd).

Samba

Samba — это Linux-имплементация сетевых протоколов, используемых для подключения к операционным системам Microsoft. В сети из Windows-хостов шеринг файлов основан на Common Internet File Systems (CIFS), который был разработан из протокола Server Message Block (SMB). Протокол Samba был разработан как бесплатный SMB-сервер для всех Unix-совместимых операционных систем, включая поддержку CIFS.

Samba прозрачно взаимодействует с CIFS таким образом, что Windows-клиенты не могут отличить, подключены ли они к WIndows или Linux-серверу. Samba предоставляет быстрый и надёжный файл и принт шеринг сервис, включая:

- Участие в Microsoft Windows Workgroup или домене как клиент, сервер или основной контроллер домена (Primary Domain Controller - PDC)
- Настройку локальных директорий как расшаренных SMB директорий, в том числе, защищенных паролем

Установка Samba

Установка Samba-сервера несколько отличается от установки nfs. Пакеты не организованы в одну группу, поэтому вы должны заранее знать, что устанавливать:

- **Samba** включает в себя базовый SMB-сервер для шаринга файлов и принтеров.
- **Samba-client** — набор утилит, который необходим, чтобы подключиться к SMB/CIFS-шарам и принтерам.
- **Samba-common** — набор общих файлов, используемых и клиентом и сервером.
- **Samba-dc** предоставляет интеграцию с Active DIrectory.
- **Samba-libs** — необходимый набор библиотек.
- **Samba-winbind** предоставляет WInbind-демон, который дает возможность Samba-серверу быть членом Microsoft-based-домена и поддержку Windows-пользователей на Linux-серверах.

- **Samba-winbind-krb5-locator** разрешает использовать локальную Kerberos-библиотеку как Samba.
- **Samba-winbind-modules** предоставляет клиентский доступ к Winbind-демону через PAM и Network Service Switch (NSS).

Samba-сервисы дают возможность взаимодействия между Microsoft WIndows и Linux-системами. Перед тем, как установить и настроить Samba, вам следует понять, как работает сеть из устройств Microsoft Windows поверх TCP/IP.

Изначально в Windows-сети каждое устройство имело уникальный хостнейм с Net-BIOS-именем, ограниченным 15 символами. Все запросы в такой сети были широковещательными, а транспорт обеспечивался протоколом NetBIOS Extended User Interface (NetBEUI), который был не маршрутизируемым. Другими словами, нельзя было обеспечить взаимодействие между разными сетями. В результате администраторы не могли построить Microsoft-сеть размером более 100-200 хостов в ней.

Чтобы обойти это ограничение, в Microsoft решили доработать NetBIOS и имплементировали его поверх TCP/IP с SMB. Далее SMB был объявлен стандартом и любой желающий может разрабатывать сервис поверх SMB, а теперь и CIFS.

Одна из фич Microsoft-сетей — возможность использования браузера для просмотра содержимого папок. Все участники сети регистрируются в выбранном мастере, который хранит всю базу данных об участниках сервиса. При этом мастер выбирается для каждого сервиса, запущенного в Microsoft-сети.

Порты, firewall, Samba

Samba, как клиент и как сервис, требует доступ по нескольким портам. Чтобы разрешить доступ к серверу, необходимо выполнить следующее:

```
[root@cifs-server ~]#firewall-cmd --get-active-zones
public interfaces:
  ens33

[root@cifs-server ~]#firewall-cmd --permanent --zone=public --add-
service=samba
[root@cifs-server ~]#firewall-cmd --reload
```

Для клиента:

```
[root@cifs-client   ~]#firewall-cmd   --get-active-
zones public interfaces: ens33

[root@cifs-client        ~]#firewall-cmd       --permanent
--add-service=samba-client
[root@cifs-client ~]#firewall-cmd --reload                          --
                                                                    zone=public
```

Чтобы посмотреть, что скрывается под сервисом Samba и Samba-client, мы можем заглянуть в директорию **/usr/lib/firewalld/services**:

```
[root@cifs-server ~]# cat /usr/lib/firewalld/services/samba.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>Samba</short>
  <description>This option allows you to access and participate in Windows file and printer sharing networks. You need the samba package installed for this option to be useful.</description>
  <port protocol="udp" port="137"/>
  <port protocol="udp" port="138"/>
  <port protocol="tcp" port="139"/>
  <port protocol="tcp" port="445"/>
  <module name="nf_conntrack_netbios_ns"/>
</service>
```

Для клиента:

```
[root@cifs-client ~]# cat /usr/lib/firewalld/services/samba-client.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>Samba Client</short>
  <description>This option allows you to access Windows file and printer sharing networks. You need the samba-client package installed for this option to be useful.</description>
  <port protocol="udp" port="137"/>
  <port protocol="udp" port="138"/>
  <module name="nf_conntrack_netbios_ns"/>
</service>
```

- 137 UDP — NetBIOS name service.
- 138 UDP — NetBIOS datagram service.
- 139 TCP — NetBIOS session service.
- 445 TCP — Samba over TCP/IP.

Так как взаимодействие между клиентом и сервером происходит в том числе через протокол UDP, который с точки зрения сети является stateless-протоколом, необходим модуль **nf_conntrack_netbios_ns**, который будет сопоставлять UDP-пакеты, пришедшие на хост и покидающие его.

Samba-демоны

Шаринг директорий и принтеров в Microsoft-сети требует несколько демонов и утилит. Для запуска Samba потребуется как минимум сам Samba-сервис (**smbd**) и NetBIOS name service (**nmbd**). Дополнительно некоторые администраторы могут захотеть запустить Winbind-сервис (**winbindd**) для преобразования юзеров и хостнеймов. Все три демона настраиваются в /etc/samba/smb.conf.

Для начала следует установить необходимые пакеты:

```
[root@cifs-server ~]# yum install -y samba samba-common samba-
winbind

Dependencies Resolved

================================================================
 Package              Arch     Version             Repository
                                                          Size
================================================================
Installing:
 samba                x86_64   4.8.3-4.el7         base   680 k
 samba-common         noarch   4.8.3-4.el7         base   206 k
 samba-winbind        x86_64   4.8.3-4.el7         base   538 k
Installing for dependencies:
 avahi-libs           x86_64   0.6.31-19.el7       base    61
 k cups-libs           x86_64   1:1.6.3-35.el7      base   357
 k libldb              x86_64   1.3.4-1.el7         base   137
 k libtalloc           x86_64   2.1.13-1.el7        base    32
 k libtdb              x86_64   1.3.15-1.el7        base    48
 k libtevent           x86_64   0.9.36-1.el7        base    36
 k libwbclient         x86_64   4.8.3-4.el7         base   109
 k pytalloc            x86_64   2.1.13-1.el7        base    17
 k samba-client-libs    x86_64   4.8.3-4.el7        base   4.8
 M samba-common-libs    x86_64   4.8.3-4.el7        base   164
 k samba-common-tools   x86_64   4.8.3-4.el7        base   448
 k samba-libs          x86_64   4.8.3-4.el7         base   276
 k samba-winbind-modules   x86_64   4.8.3-4.el7     base   115
 k

Transaction Summary
================================================================
```

Далее включаем эти демоны:

```
[root@cifs-server ~]# systemctl enable smb nmb winbind
Created symlink from /etc/systemd/system/multi-user.target.wants/smb.service to
/usr/lib/systemd/system/smb.service.
Created symlink from /etc/systemd/system/multi-user.target.wants/nmb.service to
/usr/lib/systemd/system/nmb.service.
Created symlink from /etc/systemd/system/multi-
user.target.wants/winbind.service to /usr/lib/systemd/system/winbind.service.
[root@cifs-server ~]# systemctl start smb nmb winbind
```

Проверяем статус:

```
[root@cifs-server ~]# systemctl status smb
● smb.service - Samba SMB Daemon
   Loaded: loaded (/usr/lib/systemd/system/smb.service; enabled; vendor preset:
disabled)
   Active: active (running) since Sun 2019-07-21 11:09:09 CEST; 1min 48s ago
     Docs: man:smbd(8)
           man:samba(7)
           man:smb.conf(5)
 Main PID: 11210 (smbd)
   Status: "smbd: ready to serve connections..."
   CGroup: /system.slice/smb.service
           ├─11210 /usr/sbin/smbd --foreground --no-process-g...
           ├─11214 /usr/sbin/smbd --foreground --no-process-g...
           ├─11215 /usr/sbin/smbd --foreground --no-process-g...
           └─11217 /usr/sbin/smbd --foreground --no-process-g...

Jul 21 11:09:09 cifs-server systemd[1]: Starting Samba SMB Da...
Jul 21 11:09:09 cifs-server smbd[11210]: [2019/07/21 11:09:09...
Jul 21 11:09:09 cifs-server smbd[11210]:   daemon_ready: STAT...
Jul 21 11:09:09 cifs-server systemd[1]: Started Samba SMB Dae...
Hint: Some lines were ellipsized, use -l to show in full.
[root@cifs-server ~]# systemctl status nmb
● nmb.service - Samba NMB Daemon
   Loaded: loaded (/usr/lib/systemd/system/nmb.service; enabled; vendor preset:
disabled)
   Active: active (running) since Sun 2019-07-21 11:09:09 CEST; 1min 54s ago
     Docs: man:nmbd(8)
           man:samba(7)
           man:smb.conf(5)
 Main PID: 11211 (nmbd)
   Status: "nmbd: ready to serve connections..."
   CGroup: /system.slice/nmb.service
           └─11211 /usr/sbin/nmbd --foreground --no-process-g...

Jul 21 11:09:09 cifs-server systemd[1]: Starting Samba NMB Da...
Jul 21 11:09:09 cifs-server nmbd[11211]: [2019/07/21 11:09:09...
```

```
Jul 21 11:09:09 cifs-server nmbd[11211]:   daemon_ready: STAT...
Jul 21 11:09:09 cifs-server systemd[1]: Started Samba NMB Dae...
Jul 21 11:09:43 cifs-server nmbd[11211]: [2019/07/21 11:09:43...
Jul 21 11:09:43 cifs-server nmbd[11211]:   *****
Jul 21 11:09:43 cifs-server nmbd[11211]:
Jul 21 11:09:43 cifs-server nmbd[11211]:   Samba name server ...
Jul 21 11:09:43 cifs-server nmbd[11211]:
Jul 21 11:09:43 cifs-server nmbd[11211]:   *****
Hint: Some lines were ellipsized, use -l to show in full.
[root@cifs-server ~]# systemctl status winbind
● winbind.service - Samba Winbind Daemon
     Loaded: loaded (/usr/lib/systemd/system/winbind.service; enabled; vendor
preset: disabled)
   Active: active (running) since Sun 2019-07-21 11:09:09 CEST; 2min 5s ago
     Docs: man:winbindd(8)
           man:samba(7)
           man:smb.conf(5)
 Main PID: 11212 (winbindd)
   Status: "winbindd: ready to serve connections..."
   CGroup: /system.slice/winbind.service
           ├─11212 /usr/sbin/winbindd --foreground --no-proce...
           └─11218 /usr/sbin/winbindd --foreground --no-proce...

Jul 21 11:09:09 cifs-server systemd[1]: Starting Samba Winbin...
Jul 21 11:09:09 cifs-server winbindd[11212]: [2019/07/21 11:0...
Jul 21 11:09:09 cifs-server winbindd[11212]:   initialize_win...
Jul 21 11:09:09 cifs-server winbindd[11212]: [2019/07/21 11:0...
Jul 21 11:09:09 cifs-server systemd[1]: Started Samba Winbind...
Jul 21 11:09:09 cifs-server winbindd[11212]:   daemon_ready: ...
Hint: Some lines were ellipsized, use -l to show in full.
```

Настройка samba-сервера в основном происходит через конфигурационный файл **/etc/samba/smb.conf**. Сам по себе файл достаточно большой и подразумевает некоторое понимание основ Microsoft-сетей. К счастью, /etc/samba/smb.conf включает в себя полезные комментарии и примеры.

Перед тем, как настраивать Samba, следует понять, как организован файл, какие стоят параметры по умолчанию, после чего сделать бекап /etc/samba/smb.conf и приступить к настройке.

Проверить валидность синтаксиса smb.conf можно при помощи утилиты **testparam**. Для применения изменений следует перезагрузить демон smb:

```
[root@cifs-server ~]# systemctl reload smb
```

Примеры и инструкции по настройке находятся в файле **/etc/samba/smb.conf.example** состоит из комментариев двух типов. # используется для “классических” текстовых комментариев. В основном, используются для описания фич. Второй тип коментариев ; используется для комментирования Samba

директив (возможно, вы захотите раскомментировать эти строчки в будущем, для

включения/выключения каких-либо фич)

Global Settings

Глобальная секция конфига отражает общие атрибуты для сервера:

```
#====================== Global Settings ===================================

[global]

# ----------------------- Network-Related Options -------------------------
#
# workgroup = the Windows NT domain name or workgroup name, for example,
MYGROUP.
#
# server string = the equivalent of the Windows NT Description field.
#
# netbios name = used to specify a server name that is not tied to the
hostname, #                maximum is 15 characters.
#
# interfaces = used to configure Samba to listen on multiple network
interfaces. # If you have multiple interfaces, you can use the "interfaces ="
option to
# configure which of those interfaces Samba listens on. Never omit the
localhost # interface (lo).
#
# hosts allow = the hosts allowed to connect. This option can also be used on
a # per-share basis.
#
# hosts deny = the hosts not allowed to connect. This option can also be used
on # a per-share basis.
# workgroup = MYGROUP server string = Samba
        Server Version %v

;       netbios name = MYSERVER

;       interfaces = lo eth0 192.168.12.2/24
192.168.13.2/24 ;       hosts allow = 127. 192.168.12.
192.168.13.
```

- **workgroup** — переменная workgroup может означать имя рабочей группы, но,несмотря на название, наиболее часто представляет собой домен.
- **server string** — комментарий, который показывается вместе с NetBIOS-именем в браузере.
- **netbios name** — хорошей идеей является установка имени сервера. Это имя может совпадать с хостнеймом, а может быть любым другим. Оно ограничено пятнадцатью символами. Это имя отображается либо в браузере клиента, либо при запуске net view / smbclient.
- **interfaces** — если у сервера больше одного интерфейса, можно ограничить список

интерфейсов, где будет слушаться сокет.

- **host allow** — описывает, с каким IP-адресом клиенты могут подключаться к серверу.
- **host deny** — процедура, обратная host allow. Описывает сети/адреса, с которых подключаться запрещено.
- **max protocol** — указывает максимально поддерживаемый сервером протокол.

Конфигурация логирования

```
# ------------------------- Logging Options -----------------------------
#
# log file = specify where log files are written to and how they are split.
#
# max log size = specify the maximum size log files are allowed to reach.
Log # files are rotated when they reach the size specified with "max log
size".
#

        # log files split per-machine:
        log file = /var/log/samba/log.%m
        # maximum size of 50KB per log file, then rotate:
        max log size = 50
```

Как видно, исходя из настроек, файл логов пишется для каждого хоста, который подключается к серверу. Местонахождение файла с логами:

- log file = /var/log/samba/log.%mб, %m — имя клиента.
- max log size = 50. Ограничивает размер каждого файла с логами 50 KB. Файлы, превышающие этот размер, будут ротированы.

Конфигурация автономного сервера

Настройка опций безопасности сервера может быть достаточно комплексной. В случае с отдельно стоящим сервером настройкой по умолчанию для директории является локальная база данных логинов и паролей.

```
# ----------------------- Standalone Server Options ----------------------
#
# security = the mode Samba runs in. This can be set to user,
share # (deprecated), or server (deprecated).
#
# passdb backend = the backend used to store user information in. New
# installations should use either tdbsam or ldapsam. No additional
configuration
# is required for tdbsam. The "smbpasswd" utility is available for
backwards # compatibility.
#

        security = user passdb
        backend = tdbsam
```

- значения user и server устарели. В качестве альтернативы вы можете настроить сервер как члена Active DIrectory Services (ADS); security = ads.

- Локальная база данных паролей может быть **smbpasswd** или **tdbsam**. **tdbsam** (Trivial Database Security Accounts Manager) — приоритетный выбор в случае с локальной базой (файл базы данных находится в **/var/lib/private**-директории).
- passdb backend = ldapsam, в качестве альтернативы можно настроить удаленную базу данных LDAP.

Параметры контроллера домена

```
# ----------------------- Domain Controller Options ----------------------
#
# security = must be set to user for domain controllers.
#
# passdb backend = the backend used to store user information in. New
# installations should use either tdbsam or ldapsam. No additional
configuration
# is required for tdbsam. The "smbpasswd" utility is available for
backwards # compatibility.
#
# domain master = specifies Samba to be the Domain Master Browser, allowing
# Samba to collate browse lists between subnets. Do not use the "domain master"
# option if you already have a Windows NT domain controller performing this
task.
#
# domain logons = allows Samba to provide a network logon service for
Windows # workstations.
#
# logon script = specifies a script to run at login time on the client.
These # scripts must be provided in a share named NETLOGON.
#
# logon path = specifies (with a UNC path) where user profiles are stored.
#
#
;       security = user
;       passdb backend = tdbsam

;       domain master = yes
;       domain logons = yes

        # the following login script name is determined by the machine
        name # (%m):
;       logon script = %m.bat
        # the following login script name is determined by the UNIX user used:
;       logon script = %u.bat
;       logon path = \\%L\Profiles\%u
        # use an empty path to disable profile support:
;       logon path =

        # various scripts can be used on a domain controller or a stand-
        alone # machine to add or delete corresponding UNIX accounts:
;       add user script = /usr/sbin/useradd "%u" -n -g users
;       add group script = /usr/sbin/groupadd "%g"
;         add machine script = /usr/sbin/useradd -n -c "Workstation (%u)" -M -d
```

```
/nohome -s /bin/false "%u"
;       delete user script = /usr/sbin/userdel "%u"
;       delete user from group script = /usr/sbin/userdel "%u" "%g" ;
delete group script = /usr/sbin/groupdel "%g"
```

Разрешение имен

```
#---------------------------- Name Resolution ------------------------------
#
# This section details the support for the Windows Internet Name Service
(WINS).
#
# Note: Samba can be either a WINS server or a WINS client, but not both.
#
# wins support = when set to yes, the NMBD component of Samba enables its
WINS # server. #
# wins server = tells the NMBD component of Samba to be a WINS client.
#
# wins proxy = when set to yes, Samba answers name resolution queries on
behalf # of a non WINS capable client. For this to work, there must be at
least one # WINS server on the network. The default is no.
#
# dns proxy = when set to yes, Samba attempts to resolve NetBIOS names via
DNS # nslookups.

;       wins support = yes
;       wins server = w.x.y.z
;       wins proxy = yes

;       dns proxy = yes
```

Windows Internet Name Service (WINS) функционально аналогичен DNS в Microsoft-сетях. Если вы активируете опцию **wins support = yes**, nmbd-демон включит поддержку WINS на сервере. В качестве альтернативы ваш сервер может стать клиентом удаленного WINS-сервера, если указать **wins server = w.x.y.z**. Обе опции не могут быть использованы совместно.

WINS может быть неподдерживаемой опцией у некоторых клиентов. Для таких клиентов можно включить проксирование запросов **wins proxy = yes**.

DNS proxy поможет в ситуации, когда Samba-сервер выступает WINS-сервером, но NetBIOS-имя не было зарегистрировано и требуется сделать DNS lookup.

Опции печати

Следующие настройки необходимы, чтобы расшарить принтеры при помощи Samba-сервера.

```
# ------------------------- Printing Options ---------------------------
#
# The options in this section allow you to configure a non-default
printing # system.
#
# load printers = when set you yes, the list of printers is
automatically # loaded, rather than setting them up individually.
#
# cups options = allows you to pass options to the CUPS library. Setting this
# option to raw, for example, allows you to use drivers on your Windows
clients.
#
# printcap name = used to specify an alternative printcap file.
#

        load printers = yes
        cups options = raw

;       printcap name = /etc/printcap
        # obtain a list of printers automatically on UNIX System V systems:
;       printcap name = lpstat
;       printing = cups
```

The Samba User Database

Вы можете установить идентичные имена пользователей и пароли для компьютеров под управлением Microsoft Windows и Samba с Linux. Однако это не всегда возможно, особенно когда есть уже существующие базы данных пользователей. В этом случае вы можете настроить пользователей и пароли Samba, которые соответствуют текущим именам пользователей и паролям Microsoft в вашей сети.

Самый быстрый способ настроить пользователей Samba — команда smbpasswd, предоставляемая RPM-пакетом samba-client. Помните, что вы можете создать нового пользователя Samba только из действительных учетных записей на компьютере с Linux.

Тем не менее, вы можете настроить такую учетную запись без прав входа в систему Linux. Например, следующая команда добавляет указанного пользователя без действительной оболочки входа в систему:

```
[root@cifs-server samba]# useradd testuser1 -s /sbin/nologin
```

После вы можете добавить Samba-пользователя с паролем при помощи:

```
[root@cifs-server samba]# smbpasswd -a testuser1 New SMB password:
Retype new SMB password:
Added user testuser1.
```

Расположение базы данных аутентификации зависит от значения настроек passdb. Если установлен старый формат smbpasswd, вы найдёте его в файле /etc/samba/smbpasswd. А вот tdbsam вы найдёте в файле passwd.tdb в каталоге /var/lib/samba/private. Чтобы прочитать список текущих пользователей, выполните следующую команду:

```
[root@cifs-server samba]# pdbedit -L
testuser1:1000:
```

Создание публичной директории

Для создания публично доступной директории следует использовать файл **/etc/samba/smb.conf**.

```
[PublicShare] comment = Shared
        Public Dir path =
        /var/publicshare writable
        = yes guest ok = yes
        browseable = yes
```

Как видно из настроек, шара доступна для всех и на чтение и на запись, а также доступна для браузинга.

Для защиты и контроля доступа к шаре можно использовать списки контроля доступа.

```
hosts allow = .example.com
```

Сами списки могут быть достаточно гибкими, например, можно исключить какой-либо хостнейм (в нашем случае vasya.example.com).

```
hosts allow = .example.com EXCEPT vasya.example.com
```

Кроме хостнеймов, можно использовать подсети.

```
hosts allow = 192.0.2.
или hosts allow = 192.0.2.0/255.255.255.0 Можно задать
права на папку:
```

```
chmod 1777 /var/publicshare
```

Цифра (1) перед строкой прав каталога 777 известна как sticky bit. В то время как права 777 разрешают чтение, запись и выполнение всем пользователям, sticky bit разрешает возможность удаления или переименования файлов только их владельцам.

Перед тем, как применять настройки и перезапускать сервис, необходимо проверить правильность синтаксиса утилитой testparm.

```
[root@cifs-server samba]# testparm
Load smb config files from /etc/samba/smb.conf
rlimit_max: increasing rlimit_max (1024) to minimum Windows limit (16384)
Processing section "[homes]"
Processing section "[printers]"
Processing section "[PublicShare]"
Loaded services file OK.
Server role: ROLE_STANDALONE

Press enter to see a dump of your service definitions

# Global parameters
[global] printcap name = cups
        security = USER workgroup =
        SAMBA idmap config * :
        backend = tdb cups options =
        raw


[homes] browseable = No comment =
        Home Directories inherit
        acls = Yes read only = No
        valid users = %S %D%w%S


[printers] browseable = No
        comment = All
        Printers create mask
        = 0600 path =
        /var/tmp printable =
        Yes


[PublicShare]
        comment = Shared Public
        Dir guest ok = Yes path =
        /var/publicshare read only
        = No
```

Samba-клиент

Вы можете настроить два типа клиентов через Samba. Один из них — FTP-подобный клиент, который может просматривать каталоги и получать доступ к принтерам, совместно используемым с серверов Microsoft Windows или серверов Samba в Linux / Unix. Второй добавляет поддержку команды mount для протоколов SMB и CIFS. Клиентские команды Samba доступны из RPM-пакетов samba-client и cifs-utils.

```
[root@cifs-client ~]# yum install samba-client
Dependencies Resolved

================================================================
 Package             Arch     Version           Repository
                                                          Size
================================================================
Installing: samba-client        x86_64   4.8.3-6.el7_6
updates   619 k
Installing for dependencies:
 avahi-libs          x86_64   0.6.31-19.el7     base       61 k
 cups-libs           x86_64   1:1.6.3-35.el7    base      357 k
 libarchive          x86_64   3.1.2-10.el7_2    base      318 k
 libldb              x86_64   1.3.4-1.el7       base      137 k
 libsmbclient        x86_64   4.8.3-6.el7_6     updates   134 k
 libtalloc           x86_64   2.1.13-1.el7      base       32 k
 libtdb              x86_64   1.3.15-1.el7      base       48 k
 libtevent           x86_64   0.9.36-1.el7      base       36 k
 libwbclient         x86_64   4.8.3-6.el7_6     updates   109 k
 samba-client-libs   x86_64   4.8.3-6.el7_6     updates   4.8 M
 samba-common        noarch   4.8.3-6.el7_6     updates   206 k
 samba-common-libs   x86_64   4.8.3-6.el7_6     updates   164 k

Transaction Summary
================================================================
```

Так как в наших шарах разрешен анонимный вход, можно не вводить имя пользователя и пароль при использовании **smbclient:**

```
[root@cifs-client ~]# smbclient -L 192.168.1.39 Enter SAMBA\root's password:
Anonymous login successful

        Sharename       Type      Comment
        ---------       ----      -------
```

```
        PublicShare     Disk      Shared Public Dir
        IPC$            IPC       IPC Service (Samba 4.8.3)
Reconnecting with SMB1 for workgroup listing.
Anonymous login successful

        Server               Comment
        ---------            -------

        Workgroup            Master
        ---------            -------
        SAMBA                CIFS-SERVER
```

Использование **smbclient** неудобно, и более простой вариант — монтирование Samba-директории.

Для начала установим **cifs-utils:**

```
[root@cifs-client ~]# yum install cifs-utils

Dependencies Resolved

================================================================
 Package         Arch        Version            Repository
                                                        Size
================================================================
Installing: cifs-utils      x86_64      6.2-10.el7         base
85 k

Transaction Summary
================================================================
Install  1 Package

Total download size: 85 k
Installed size: 175 k
```

Затем используем mount-команду с опцией cifs:

```
[root@cifs-client      ~]#   mount.cifs  //192.168.1.39/PublicShare    /mnt/ -o
username=testuser1
Password for testuser1@//192.168.1.39/PublicShare:  **********
[root@cifs-client ~]#
[root@cifs-client ~]# mount | grep 39
//192.168.1.39/PublicShare          on          /mnt          type          cifs
(rw,relatime,vers=default,cache=strict,username=testuser1,domain=,uid=0,noforce
u
id,gid=0,noforcegid,addr=192.168.1.39,file_mode=0755,dir_mode=0755,soft,nounix,
s erverino,mapposix,rsize=1048576,wsize=1048576,echo_interval=60,actimeo=1)
```

iSCSI

iSCSI — первые буквы от словосочетания Internet Small Computer System Interface. Иногда это буквосочетание расшифровывают как Internet SCSI или IP SCSI, и, несмотря на то что такие интерпретации не полностью совпадают с первоначальной, они вполне имеют право на жизнь, так как

весьма точно описывают суть iSCSI. Это протокол из стека TCP/IP для подключения внешних сетевых систем хранения данных в режиме блочного доступа.

В основу метода заложена трансляция команд SCSI посредством IP-сети. В процессе работы используются порты TCP/IP, по умолчанию — 860 и 3260. В принципе, iSCSI —это своего рода транспорт вроде эскалатора для перемещения SCSI-инструкций и данных через внешнее сетевое подключение. Конечная реализация представляет собой среду для эмуляции локальной шины SCSI посредством внешней сети Ethernet и т. д.

iSCSI строится на двух наиболее широко используемых протоколах:

- SCSI — протоколе обмена блоками данных между компьютером и хранилищем.
- IP — сетевом транспортном протоколе, наиболее широко применяющемся сегодня в корпоративных сетях Ethernet.

Использование стандартного набора команд SCSI упрощает совместимость с существующими операционными системами и приложениями. Использование TCP/IP обеспечивает передачу команд SCSI в глобальном масштабе.

Архитектура обычного SCSI базируется на клиент-серверной модели. «Клиент», которым может быть, например, физический сервер или рабочая станция, инициирует запросы на считывание или запись данных с исполнителя — «сервера», в роли которого, как правило, выступает система хранения данных (СХД). Команды, которые выдает «клиент» и обрабатывает «сервер», помещаются в блок описания команды (Command Descriptor Block, CDB).

CDB — структура, с помощью которой приложение-клиент направляет команды устройству-серверу. «Сервер» выполняет команду, а окончание ее выполнения обозначается специальным сигналом. Инкапсуляция и надежная доставка CDB-транзакций между инициаторами и исполнителями через TCP/IP сеть и есть главная задача iSCSI, причем ее приходится осуществлять в нетрадиционной для SCSI, потенциально ненадёжной среде IP-сетей.

Протокол iSCSI контролирует передачи блоков данных и обеспечивает подтверждение достоверности завершения операции ввода/вывода, что в свою очередь обеспечивается через одно или несколько TCP-соединений.
В отличие от многих других протоколов (FCIP, FCoE и так далее), являющихся, по сути, ответвлением от Fibre Channel, протокол iSCSI является независимой реализацией и представляет собой разработанный с нуля стандарт для работы через TCP/IP.

iSCSi target и iSCSi initiator

Для организации любой сети хранения данных требуется три составляющие: система хранения данных, клиентская часть и среда передачи данных. В случае iSCSI для описания первых двух составляющих применяются термины **target** и **initiator** соответственно.

- **Target**, или целевое устройство, — основа системы хранения, может иметь программную реализацию в чистом виде, программно-аппаратную и полностью аппаратную.
- **Initiator** — модуль, чаще всего программа, реже аппаратное решение со своим firmware, позволяющее создавать (инициировать) соединение и обеспечивающее нужный функционал на стороне клиента — передачу SCSI-команд и данных по IP-сети.

Адресация iSCSI target

Для успешной безошибочной работы системы хранения, подключенные к сети, обязаны иметь уникальный сетевой адрес. Например, сети хранения данных на базе протокола Fibre Channel используют специальные адреса WWWN. SAN на основе iSCSI также имеют свою собственную систему адресации — **IQN** (iSCSI Qualified Name).

Каждый такой адрес — уникальный идентификатор, служащий для точного определения устройств хранения. Формат записи: **iqn.2019–10.com.example:storage:whatever-sn-a978123440.**

Что это означает:

- **iqn** — префикс, указывающий на принадлежность адреса к формату IQN.
- Далее следует указатель даты вида yyyy-mm (год-месяц), чаще всего указывают дату создания таргета. > Зарезервированное доменное имя, чаще всего вендора оборудования.
- После двоеточия следует собственно уникальный ID iSCSI target.

Эта система довольно удобна. Прочитав IQN, можно легко получить дополнительную информацию о времени создания, типе оборудования и/или вендоре.

По аналогии с Fibre Channel, служба iSNS (Internet Storage Name Service) позволяет управлять в том числе и сетями iSCSI. Это дает возможность использовать iSNS в роли единой централизованной точки входа для работы SAN.

Управление именами и адресами

Так как iSCSI-устройства — участники IP-сети, они имеют индивидуальные сетевые сущности (Network Entity). Каждая из них может содержать один или несколько iSCSI-узлов.
iSCSI-узел — идентификатор SCSI-устройств, доступных через сеть. Каждый iSCSI-узел имеет уникальное имя (длиной до 255 байт), которое формируется по правилам, принятым для обозначения узлов в Интернете (например, fqn.com.ustar.storage.itdepartment.161) — это IQN.

В то же время в процессе контроля и передачи данных между устройствами удобнее пользоваться комбинацией IP-адреса и TCP-порта, которые обеспечиваются сетевым порталом (Network Portal). iSCSI-протокол в дополнение к iSCSI-именам обеспечивает поддержку псевдонимов, которые, как правило, отображаются в системах администрирования для удобства идентификации и управления администраторами системы.

Управление сеансом

iSCSI-сессия состоит из фаз аутентификации (Login Phase) и фазы обмена (Full Feature Phase), которая завершается специальной командой.

Фаза аутентификации используется, чтобы согласовать разнообразные параметры между двумя сетевыми сущностями и подтвердить право доступа инициатора. После процедуры аутентификации iSCSI-сессия переходит к фазе обмена. Если было установлено больше одного TCP-соединения, iSCSI требует, чтобы каждая пара «команда/ответ» также проходила через одно TCP-соединение. Такая процедура гарантирует, что каждая отдельная команда считывания или записи будет выполняться без дополнительного отслеживания прохождения каждого запроса по разным потокам. Однако разные транзакции могут одновременно передаваться через разные TCP-соединения в рамках одной сессии.

В завершение транзакции инициатор передает и принимает последние данные, а исполнитель отправляет ответ, который подтверждает успешную передачу данных.

Если нужно закрыть сессию, используется команда iSCSI logout, которая передает информацию о причинах завершения. Она также может передать информацию о том, какое соединение следует закрыть в случае возникновения ошибки соединения, чтобы закрыть проблемные TCP-связи.

Обработка ошибок

В связи с высокой вероятностью возникновения ошибок при передаче данных в некоторых типах IP-сетей, особенно в WAN-реализациях (где также может функционировать iSCSI), протокол предусматривает ряд мероприятий по обработке ошибок.

Чтобы обработка ошибок и восстановление после сбоев функционировали корректно, как инициатор, так и исполнитель должны иметь возможность буферизации команд до момента их подтверждения. Каждое конечное устройство должно иметь возможность выборочно восстановить утраченный или испорченный PDU в рамках транзакции для восстановления передачи данных.

Иерархия системы обработки ошибок и восстановление после сбоев в iSCSI включает:

1. На самом низком уровне — определение ошибки и восстановление данных на уровне SCSI-задачи, например, повторение передачи утраченного или поврежденного PDU.

2. На следующем уровне — в TCP-соединении, которое передает SCSI-задачу, может произойти ошибка, т. е., TCP-соединение может повредиться. В этом случае система пытается его восстановить.
3. И наконец, сама iSCSI-сессия может «испортиться». Терминация и восстановление сессии, как правило, не требуется, если восстановление корректно отрабатывается на других уровнях, однако может произойти обратное. Такая ситуация требует закрытия всех TCP-соединений, завершения всех задач, недовыполненных SCSI-команд и перезапуска сессии с повторной аутентификацией.

Безопасность

В связи с использованием iSCSI в сетях, где возможен несанкционированный доступ к данным, спецификация предусматривает возможность использования разнообразных методов для повышения безопасности. Такие средства шифрования, как IPSec, которые используют нижние уровни, не требуют дополнительного согласования, так как прозрачны для верхних уровней, в том числе для iSCSI. Для аутентификации могут использоваться разнообразные решения, например Kerberos или обмен частными ключами; репозиторием ключей может быть iSNS-сервер.

Варианты реализации iSCSI target

- **Программная реализация**. Пример — программный продукт StarWind iSCSI Target Software, который обеспечивает реализацию iSCSI target на обычных серверах под управлением операционных систем семейства MS Windows. Достаточно только установить программу, провести небольшие настройки, и готово блочное хранилище начального уровня для подключения к другому серверу.
- **Аппаратная реализация**. Пример — специализированные сетевые адаптеры, с особыми прошивкой, интерфейсами, чипами и Firmware, которые берут на себя большую часть функций обработки трафика.
- **Программно-аппаратная реализация** — своего рода компромиссное решение, например обычный сервер на базе платформы Intel x86_64, но со специализированными сетевыми адаптерами (TOE) и адаптированной операционной системой, к примеру, NexentaStor, позволяющей организовать iSCSI target, что называется, «сразу из коробки».

iSCSI HBA

Иногда можно встретить устойчивое выражение iSCSI HBA (Host Bus Adapter). Речь идет о специальных аппаратных сетевых модулях, позволяющих разгрузить процессор, передав часть функционала сетевому адаптеру.

Разделяют два типа таких устройств:

- TCP Offload Engines, сокращенно TOE. Эти устройства можно встретить там, где необходимо увеличить производительность и одновременно снизить нагрузку на общую систему (процессор и так далее). Данное устройство способно взять на себя только операции по поддержке TC/IP, но не способно использовать все остальные возможности по увеличению производительности iSCSI-систем.
- Full offload iSCSI HBA — комплексное решение, включающее передачу выполнения функций по поддержке TCP/IP и iSCSI на устройство. Это считается лучшим выбором по обеспечению производительности, но стоит, разумеется, дороже TOE.

Многие современные сетевые адаптеры 10 Gigabit Ethernet включают в себя поддержку протокола iSCSI. Поэтому при выборе сетевого адаптера для построения iSCSI SAN следует ориентироваться не только на цену комплектующих, но и на дополнительный функционал. Соответствующие характеристики стоит уточнить на сайте производителя оборудования.

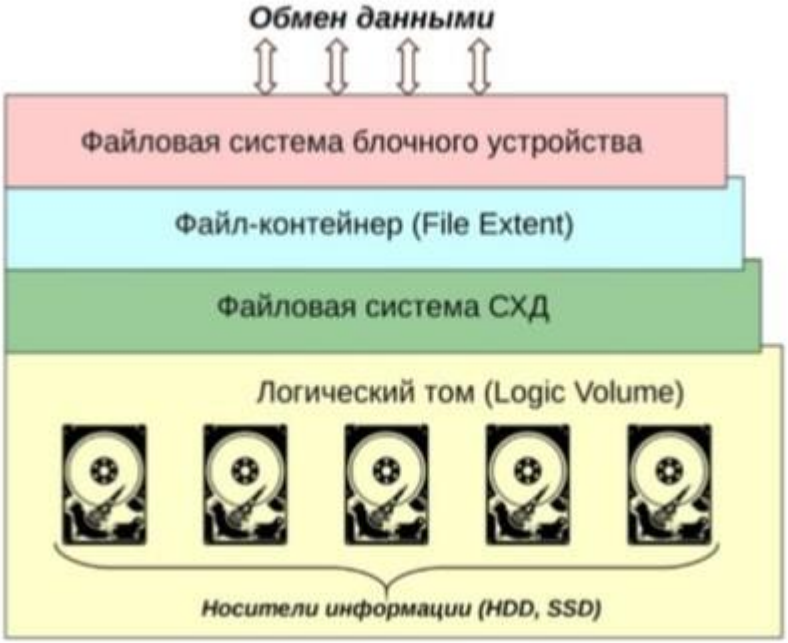
Различия File Extent и Device Extent

Помимо систем передачи трафика, есть различные подходы при сохранении данных непосредственно внутри хранилища. Участок системы хранения, отвечающий за размещение данных, доступ к которым осуществляется по протоколу iSCSI, называется Extent.

File Extent, или файл-контейнер

Этот метод встречается наиболее часто в силу достаточно простой реализации. Его суть — в использовании специального файла большого размера, в котором, как в контейнере, размещаются данные клиента. Наиболее близкий аналог — виртуальный диск (например, создаваемый системой виртуализации), который доступен как очередной жёсткий диск или съемный носитель. Другие аналоги — архивный файл, в который данные записываются «на лету», или файл-контейнер, создаваемый программой шифрования данных в качестве защищённого хранилища.

При таком подходе физический диск или дисковый массив форматируется под определенную файловую систему, в которой создаётся огромный файл-контейнер, внутри его — своя внутренняя файловая система, и уже на ней размещаются данные клиента (см. рис. 1).



Разумеется, когда создаётся каждый новый слой, определённая часть дискового объёма расходуется на запись служебной информации. В итоге конечный объем iSCSI на базе File Extent будет всегда меньше объёма логического тома, при этом общие потери могут быть весьма ощутимы.

Именно так работает большинство реализаций iSCSI target. Такой метод хранения довольно рискованный. Малейшая ошибка файловой системы при записи этого громадного файла — и все данные на нём будут утеряны. Подобные ошибки легко возникают при несанкционированной холодной перезагрузке, некорректной работе RAID-контроллера (особенно это касается RAID-контроллеров, встроенных в материнскую плату) и так далее. Не спасут проверки ни программой типа CheckDisk, ни каким-либо другим способом. Вся надежда только на своевременно созданную резервную копию.

Помимо невысокой защищенности и расхода свободного пространства, такой метод не слишком производителен. Метод работы «все-в-один-файл» — прямой аналог работы программы-архиватора.

Накладные расходы также сопоставимы с бесконечной работой программы-архиватора в активном режиме.

Device Extent

Этот метод более прост и экономичен. Из названия следует, что для сохранения используется не отдельный файл, а целиком все устройство. В таком случае нет нужды создавать своеобразную «матрешку»: внешняя файловая система → файл-контейнер → внутренняя файловая система. Вместо этого данные пишутся напрямую на дисковый том в RAW-формате. Это позволяет значительно снизить накладные расходы и избежать потенциальных ошибок, например, из-за «холодной перезагрузки».
Device Extent позволяет обеспечить большую производительность при передаче данных, а также избежать множества проблем, связанных с особенностями работы той или иной операционной

системы, конкретной реализацией iSCSI-инициатора и так далее. Разумеется, все эти преимущества будут доступны, если есть подходящий драйвер для аппаратной реализации iSCSI target. В противном случае устройство просто не будет работать.

Device Extent можно встретить в системах на платформе BSD — FreeBSD и ее производных: FreeNAS и NAS4Free.

Обеспечение безопасности при построении iSCSI SAN

Поклонники протокола Fibre Channel при обсуждении вопросов реализации безопасности подключений обязательно вспоминают Zoning — механизм, присутствующий в сетях FC. Аналогичные механизмы существуют и в iSCSI SAN.

- **Ограничение доступа по сети средствами iSCSI target.** Практически во всех реализациях iSCSI target можно программно ограничить доступ со всех адресов, за исключением небольшой группы серверов, которые нуждаются в ресурсах СХД. Метод можно сравнить с software zoning в Fibre Channel, когда в качестве атрибута используется адрес порта (устройства): IP-адрес для iSCSI или WWWN для Fibre Channel.
- **Ограничение доступа по сети внешними средствами**. Еще одна возможность — использование внешних систем для ограничения сетевого доступа. Так, в большинстве случаев сети iSCSI строятся на базе Ethernet, выделение некоторых сегментов посредством VLAN — хорошая практика для обособления iSCSI SAN, эффективно защищающая от несанкционированного доступа. В принципе, VLAN для iSCSI можно сравнить с Hardware Zoning для Fibre Channel. И в одном, и в другом случае ограничение доступа производится исходя из портов, к которым подключаются устройства.
- **Проверка подлинности CHAP**. Чаще всего для проверки легитимности подключения iSCSI initiator с iSCSI target применяется протокол CHAP (Challenge Handshake Authentication Protocol). Основу метода составляет совместное использование клиентом и сервером секретного ключа (аналогичного паролю).

Проверка подлинности с использованием шифрования

Помимо вышеописанных способов — аутентификации и ограничения доступа по сети, — для обеспечения повышенной безопасности при работе с iSCSI можно использовать различные виды шифрования. Стоит отметить, что вовсе не обязательно выбирать какой-либо один метод проверки подлинности. Например, можно использовать и шифрование, и проверку подлинности при помощи CHAP или RADIUS.

Наиболее известный метод — использование протокола IPsec, выполняющего принудительную проверку подлинности и шифрование данных на уровне IP-пакетов. При использовании IPsec все IP-

пакеты подвергаются шифрованию и проверке. Соответственно, все участники сетевого обмена должны иметь общий ключ для проверки подлинности друг друга и шифрования пакетов.

Также неплохо зарекомендовала себя возможность шифрования iSCSI-ресурсов как дисковых разделов. Такой том можно активировать только через специальную программу-агента, с обязательным вводом пароля и подключением соответствующего сертификата.

Области применения iSCSI

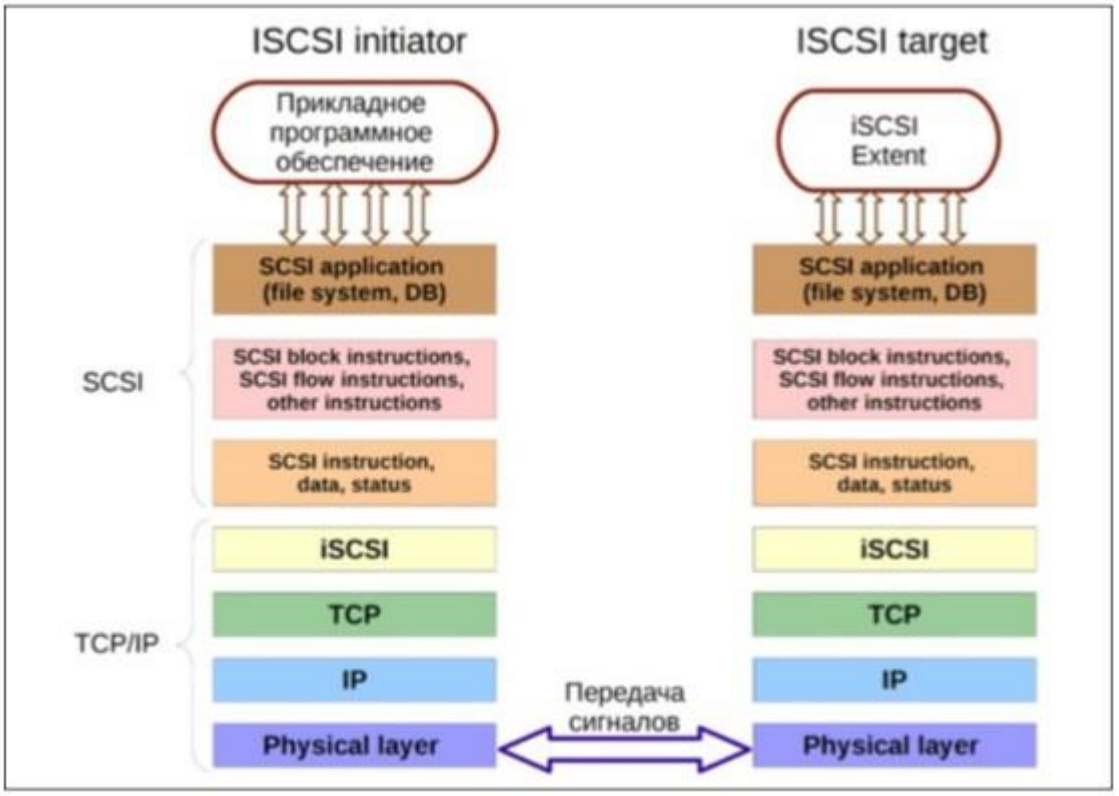
Области применения устройств хранения на базе iSCSI такие же, что и для Fibre Channel, как, впрочем, для любых других типов СХД, предоставляющих ресурсы в режиме блочного доступа.

Однако до недавнего времени из-за невысокой пропускной способности сети Gigabit Ethernet применение iSCSI было довольно ограничено. Ситуация кардинально поменялась с выходом стандарта 10 Gigabit Ethernet и началом массового выпуска соответствующего оборудования. iSCSI традиционно используется для систем удаленной загрузки, резервного копирования, создания систем хранения класса C.

В то же время неплохие возможности сетевого оборудования 10 Gigabit Ethernet позволяют использовать iSCSI SAN и при построении систем виртуализации, и для хранения баз данных, словом, для всех задач, где раньше господствовал стандарт Fibre Channel.

С какой скоростью работает iSCSI?

В отличие от внутренней шины SCSI, которая обеспечивает непосредственный доступ к устройствам, передача пакетов iSCSI происходит посредством потенциально ненадежного сетевого соединения. Для обеспечения стабильной работы, контроля обмена данными и SCSI-командами в данных условиях в работе протокола iSCSI применяется избыточность. Эта избыточность выражается в передаче дополнительной служебной информации, которая используется для мониторинга блочной передачи, проверки корректности завершения операций ввода/вывода и обработки ошибок. Также служебная информация необходима для системы идентификации устройства посредством соответствующих имен. Еще одна задача, решаемая при обмене данными, — обеспечение безопасности. Разумеется, все это вкупе с процессом инкапсуляции-деинкапсуляции ведет к дополнительным накладным расходам.



Как ускорить работу iSCSI SAN?

Используйте выделенный коммутатор. При работе iSCSI target в общей сети коммутаторы, помимо доступа к СХД, вынуждены обслуживать множество конкурентных транзакций, например между серверами и офисными компьютерами, что снижает фактическую скорость сетевого обмена. Также выделенный коммутатор — отличная мера безопасности (см. ниже).

Не используйте излишние средства безопасности. Чем больше механизмов безопасности используется, тем медленнее работает СХД в целом. Да, современные iSCSI-хранилища позволяют применять одновременно программное ограничение сетевого доступа, двунаправленную аутентификацию и шифрование IPsec. Но насколько все это необходимо в обычной ситуации? При решении простых задач достаточно выделенного коммутатора.

Используйте самую быструю сеть. Однако следует помнить: самая скоростная сеть будет бесполезна, если используются другие комплектующие слабой или устаревшей конфигурации, например старенький процессор или отживший контроллер дисковой подсистемы.

Найдите узкое место и проведите модернизацию оборудования СХД (iSCSI target). Не забывайте про возможности применения сетевых адаптеров со встроенными функциями поддержки iSCSI.

По возможности используйте Device Extent для снижения накладных расходов при работе с дисковой подсистемой iSCSI target.

Следите за свободным пространством. Избегайте заполнения подключаемых iSCSI-томов более чем на 75-80% от их объема во избежание деградации производительности. Помните, любая система хранения, будь то жесткий диск или SAN-хранилище, после превышения указанного предела будет работать медленнее.

Не создавайте очень больших дисковых томов. Большие дисковые разделы, подключаемые по сети в режиме блочного доступа, могут вызывать падение производительности из-за проблем в индексации, поиске и размещении информации на дисковом разделе. Также следует понимать, что при пропадании сети возможно появление ошибок на файловой системе подключаемых блочных ресурсов. Проверить огромный том на ошибки может оказаться крайне сложно.

Использование стандартов и технологий iSCSI позволяет быстро подключить систему хранения данных в качестве устройств блочного доступа. Однако при дальнейшем развитии IT-инфраструктуры необходимы определенные финансовые и технические ресурсы для обеспечения надежности и приемлемой скорости передачи данных.

Установка iSCSI target на Linux

Сервер Linux «из коробки» может быть как iSCSI-инициатором, так и iSCSI-таргетом. Рассмотрим для начала установку iSCSI target. За это отвечает пакет targetcli:

```
[root@iscsi-target ~]# yum install targetcli
Dependencies Resolved

================================================================================
 Package                  Arch        Version                Repository
                                                                             Size
================================================================================
Installing: targetcli               noarch      2.1.fb49-1.el7        base
68 k
Installing for dependencies:
libnl                       x86_64       1.1.4-3.el7            base       128 k
pyparsing              noarch      1.5.6-9.el7           base       94 k python-
configshell      noarch      1:1.1.fb25-1.el7      base       68 k python-ethtool
x86_64      0.8-8.el7             base       34 k python-kmod             x86_64
0.9-4.el7            base       57 k python-rtslib          noarch      2.1.fb69-
3.el7         base       102 k python-six                noarch       1.9.0-2.el7
base        29 k python-urwid             x86_64      1.1.1-3.el7            base
654 k

Transaction Summary
================================================================================
```

В пакет входит утилита targetcli, которая запускает удобный для пользователя text-based UI-конфигурации, где шаг за шагом можно будет настроить iSCSI target. После запуска оболочки targetcli введите команду ls для отображения текущих настроек.

```
[root@iscsi-target ~]# targetcli
targetcli shell version 2.1.fb49
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/> ls o- /
.......................................................... [...] o-
backstores ............................................... [...] |
o- block ..................................... [Storage Objects: 0]
  | o- fileio ..................................... [Storage Objects: 0]
  | o- pscsi ...................................... [Storage Objects:
0] | o- ramdisk .................................... [Storage Objects:
0] o- iscsi ................................................ [Targets:
0] o- loopback ............................................. [Targets:
0] />
```

Из оболочки targetcli вы можете переходить к различным разделам конфигурации с помощью команды cd, так же как и в bash. Команда ls отображает содержимое текущего раздела, а help — отдельный экран контекстной справки. Также вы можете использовать TAB для заполнения частично введенных команд или аргументов.

В качестве бэкенда хранения данных клиентов будем использовать отдельный жесткий диск, в нашем случае sdb:

```
[root@iscsi-target ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE
MOUNTPOINT sda               8:0    0   20G  0
disk
├─sda1            8:1    0    1G  0 part /boot
└─sda2            8:2    0   19G  0 part
  ├─centos-root 253:0    0   17G  0 lvm  /
  └─centos-swap 253:1    0    2G  0 lvm  [SWAP]
sdb                8:16   0    5G  0 disk sr0
11:0    1  918M  0 rom
[root@iscsi-target ~]#
```

Создадим отдельный LVM-раздел размером 1G для использования его в качестве iSCSI LUN:

```
[root@iscsi-target ~]# pvcreate /dev/sdb
  Physical volume "/dev/sdb" successfully created.
[root@iscsi-target ~]# vgcreate centos-target /dev/sdb
  Volume group "centos-target" successfully created
[root@iscsi-target  ~]#  lvcreate  -L  1G  -n  backstore  centos-target
  Logical volume "backstore" created.
[root@iscsi-target ~]# lvs
   LV          VG              Attr        LSize     Pool Origin Data%
Cpy%Sync Convert root      centos        -
  wi-ao---- <17.00g swap      centos
  -wi-ao----   2.00g
                                                                    Meta%  Move Log
 backstore centos-target -wi-a-----   1.00g
[root@iscsi-target ~]#
```

В качестве бэкенда для хранения данных клиентов может выступать не только блочное устройство, например **sd***, но и файл (**fileio**) или оперативная память (**ramdisk**).

Мы же добавили в виртуальную машину диск, создали поверх него LVM-раздел, и теперь настало время вернуться назад в **targetcli**, чтобы создать блочный backstore по имени disk1.

```
[root@iscsi-target ~]# targetcli
targetcli shell version 2.1.fb49
Copyright 2011-2013 by Datera, Inc and others.
For help on commands, type 'help'.

/> cd backstores/block
/backstores/block> create disk1 /dev/centos-target/backstore
Created block storage object disk1 using /dev/centos-target/backstore.
```

Создаем IQN для сервера

IQN должен быть создан в соответствии со специальными правилами. В нашем случае это будет: **iqn.2019-10.com.example:server1-disk1**

```
/backstores/block> cd /iscsi
/iscsi>    create    iqn.2019-10.com.example:server1-disk1
Created target iqn.2019-10.com.example:server1-disk1.
Created TPG 1.
Global pref auto_add_default_portal=true
Created default portal listening on all IPs (0.0.0.0), port
3260. /iscsi>
```

Как видно из логов, после создания IQN сервер начал слушать TCP порт 3260 на всех имеющихся у него IP-адресах (0.0.0.0). Это можно будет поменять и указать, на каким конкретно интерфейсе запускать iSCSI target.

Target Portal Group (TPG)

Настройки контроля доступа для созданного IQN находятся в новой секции конфига TGP и специфичны для каждого созданного iSCSI target.

```
/iscsi> ls
o- iscsi ..................................................... [Targets: 1] o-
  iqn.2019-10.com.example:server1-disk1 ..................... [TPGs: 1] o-
  tpg1 ..................................... [no-gen-acls, no-auth] o- acls
  ..................................................... [ACLs: 0] o- luns
  ..................................................... [LUNs: 0] o- portals
  ..................................... [Portals: 1] o- 0.0.0.0:3260
  ..................................... [OK] /iscsi>
```

Настроим список контроля доступа, чтобы только один конкретный клиент имел доступ к порталу, а также создадим для него виртуальный диск, который называется LUN (Logical Unit Number).

ACL

Так как настройка ACL специфична для каждого target IQN, мы должны продвинуться дальше по иерархии в настройку конкретного IQN: cd iqn.2019-10.com.example:server1-disk1/tpg1/acls.

Разрешим доступ только одного клиента с initiator IQN — iqn.2019-10.com.example:client.

```
/iscsi> cd iqn.2019-10.com.example:server1-disk1/tpg1/acls
/iscsi/iqn.20...sk1/tpg1/acls> create iqn.2019-10.com.example:client
Created Node ACL for iqn.2019-10.com.example:client
/iscsi/iqn.20...sk1/tpg1/acls> cd /iscsi/
/iscsi> ls
o- iscsi ..... [Targets: 1] o-
iqn.2019-10.com.example:server1-disk1 ..... [TPGs: 1] o- tpg1
..... [no-gen-acls, no-auth] o- acls
..... [ACLs: 1] | o- iqn.2019-
10.com.example:client ..... [Mapped LUNs: 0] o- luns
..... [LUNs: 0] o- portals
..... [Portals: 1] o- 0.0.0.0:3260
..... [OK] /iscsi>
```

LUN

Чтобы создать логический диск, видимый как блочное устройство у клиента, мы должны перейти в luns внутри настроек портала и указать там ранее добавленное в backstores устройство.

```
/iscsi>          cd          iqn.2019-10.com.example:server1-disk1/tpg1/luns
/iscsi/iqn.20...sk1/tpg1/luns> create /backstores/block/disk1 Created LUN 0.
Created LUN 0->0 mapping in node ACL iqn.2019-10.com.example:client
/iscsi/iqn.20...sk1/tpg1/luns> cd /iscsi/
/iscsi> ls
o- iscsi ..... [Targets: 1] o- iqn.2019-
  10.com.example:server1-disk1 ..... [TPGs: 1] o- tpg1
  ..... [no-gen-acls, no-auth] o- acls
  ..... [ACLs: 1]
```

```
      | o- iqn.2019-10.com.example:client ..... [Mapped LUNs:
     1] |   o- mapped_lun0 ..... [lun0 block/disk1
     (rw)] o- luns .....
     [LUNs: 1]
     | o- lun0  [block/disk1 (/dev/centos-target/backstore)
(default_tg_pt_gp)] o- portals .....
[Portals: 1] o- 0.0.0.0:3260 ..... [OK]
/iscsi>
```

Ограничим подсеть используемого портала, так как обычно подсеть для iSCSI не маршрутизируемая. Для этого удалим автоматически созданный 0.0.0.0, а затем назначим на портал адрес, который находится на интерфейсе ens33 — 192.168.1.41.

```
/iscsi> cd iqn.2019-10.com.example:server1-disk1/tpg1/
/iscsi/iqn.20...r1-disk1/tpg1> cd portals/
/iscsi/iqn.20.../tpg1/portals> delete 0.0.0.0 3260
/iscsi/iqn.20.../tpg1/portals> create 192.168.1.41 3260
/iscsi/iqn.20.../tpg1/portals> cd /iscsi/
/iscsi> ls
o- iscsi ..................................................... [Targets:
1] o- iqn.2019-10.com.example:server1-disk1 ..................... [TPGs:
1] o- tpg1 ....................................... [no-gen-acls, no-
auth] o- acls ..................................................... [ACLs:
1] | o- iqn.2019-10.com.example:client ..... [Mapped LUNs: 1]
|   o- mapped_lun0 ............................. [lun0 block/disk1 (rw)] o-
luns ..................................................... [LUNs: 1] | o-
lun0 . [block/disk1 (/dev/centos-target/backstore) (default_tg_pt_gp)] o-
portals ................................................ [Portals: 1] o-
192.168.1.41:3260 ..................................... [OK] /iscsi>
```

Теперь проверим все настройки и выйдем при помощи Exit для сохранения изменений. После чего следует включить сервис targetcli и разрешить порт 3260 в **firewalld**.

```
/> ls
o- / ..........................................................................
  [...] o- backstores ..............................................................
  [...] | o- block ..................................... [Storage
  Objects: 1]
  | | o- disk1 ..... [/dev/centos-target/backstore (1.0GiB) 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: 1]
  | o- iqn.2019-10.com.example:server1-disk1 ..................... [TPGs: 1]
  |   o- tpg1 ....................................... [no-gen-acls, no-auth]
  |     o- acls ..................................................... [ACLs: 1]
  |     | o- iqn.2019-10.com.example:client ................... [Mapped LUNs: 1]
```

```
  |     |   o- mapped_lun0 ............................. [lun0 block/disk1 (rw)]
  |     o- luns ..................................................... [LUNs: 1]
 |     | o- lun0  [block/disk1 (/dev/centos-target/backstore) (default_tg_pt_gp)]
  |     o- portals ............................................... [Portals:
  1] |       o- 192.168.1.41:3260 ..........................................
  [OK] o- loopback ..................................................
  [Targets: 0]
/>/> exit
Global pref auto_save_on_exit=true
Configuration saved to /etc/target/saveconfig.json
[root@iscsi-target ~]# systemctl enable target
Created symlink from /etc/systemd/system/multi-user.target.wants/target.service
/usr/lib/systemd/system/target.service.
[root@iscsi-target ~]# systemctl start target
[root@iscsi-target ~]# firewall-cmd --permanent --add-port=3260/tcp
success
[root@iscsi-target ~]# firewall-cmd --reload
success                                                                    to
```

Установка iSCSI initiator на Linux

iSCSI-клиент должен иметь набор софта для подключения к iSCSI target. За это отвечает пакет **iscsi-initiator-utils**.

```
[root@iscis-initiator ~]# yum install iscsi-initiator-utils
Dependencies Resolved

================================================================================
 == Package                          Arch      Version
 Repository
                                                                          Size
================================================================================
== Installing: iscsi-initiator-utils             x86_64    6.2.0.874-11.el7
base    422 k
Installing for dependencies: iscsi-initiator-utils-iscsiuio    x86_64
 6.2.0.874-11.el7      base     92 k

Transaction Summary
================================================================================
==
```

Теперь можно задать имя инициатора в /etc/iscsi/initiatorname.iscsi. Мы уже указали имя инициатора в настройках портала — **iqn.2019-10.com.example:client**.

```
[root@iscis-initiator ~]# vim /etc/iscsi/initiatorname.iscsi
InitiatorName=iqn.2019-10.com.example:client
```

Включаем службу iscsi.

```
[root@iscis-initiator ~]# systemctl start iscsi
[root@iscis-initiator ~]# systemctl enable iscsi
```

Пытаемся обнаружить настроенный портал (iSCSI target) с IP-адресом 192.168.1.41. Для этого используется утилита iscsiadm:

```
[root@iscis-initiator ~]# iscsiadm -m discoverydb -t st -p 192.168.1.41 -D
192.168.1.41:3260,1 iqn.2019-10.com.example:server1-disk1
```

Как видно из вывода команды, инициатор в состоянии обнаружить таргет. Настало время подключиться:

```
[root@iscis-initiator ~]# iscsiadm -m node -T iqn.2019-10.com.example:server1-disk1 -l
Logging in to [iface: default, target: iqn.2019-10.com.example:server1-disk1, portal:
192.168.1.41,3260] (multiple)
Login to [iface: default, target: iqn.2019-10.com.example:server1-disk1, portal:
192.168.1.41,3260] successful.
```

Проверяем, что диск виден:

```
[root@iscis-initiator ~]# lsblk
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0   20G  0 disk
├─sda1            8:1    0    1G  0 part /boot
└─sda2            8:2    0   19G  0 part
  ├─centos-root 253:0    0   17G  0 lvm  /
  └─centos-swap 253:1    0    2G  0 lvm  [SWAP]
sdb                8:16   0    1G  0 disk sr0
11:0    1  918M  0 rom

[root@iscis-initiator ~]# dmesg | tail -n 15

[ 5456.297332] Loading iSCSI transport class v2.0-
870. [ 5456.307933] iscsi: registered transport (tcp)
[ 5574.290497] scsi host3: iSCSI Initiator over TCP/IP
[ 5574.295868] scsi 3:0:0:0: Direct-Access      LIO-ORG disk1              4.0 PQ: 0
ANSI: 5
[ 5574.309553] scsi 3:0:0:0: alua: supports implicit and explicit TPGS
[ 5574.309557] scsi 3:0:0:0: alua: device naa.600140557591f3c00d94916a0e8f4f15 port
group 0 rel port 1
[ 5574.309559] scsi 3:0:0:0: alua: Attached
[ 5574.310069] sd 3:0:0:0: Attached scsi generic sg2 type 0
[ 5574.310238] sd 3:0:0:0: [sdb] 2097152 512-byte logical blocks: (1.07 GB/1.00 GiB)
[ 5574.313442] sd 3:0:0:0: [sdb] Write Protect is off
```

```
[ 5574.313445] sd 3:0:0:0: [sdb] Mode Sense: 43 00 00 08
[ 5574.314066] sd 3:0:0:0: [sdb] Write cache: disabled, read cache: enabled,
doesn't support DPO or FUA
[ 5574.315065] sd 3:0:0:0: alua: transition timeout set to 60 seconds
[ 5574.315068] sd 3:0:0:0: alua: port group 00 state A non-preferred supports
TOlUSNA [ 5574.322049] sd 3:0:0:0: [sdb] Attached SCSI disk
```

Теперь можно добавить на устройство файловую систему и примонтировать сам диск.

```
[root@iscis-initiator ~]# mkfs.ext4 /dev/sdb
mke2fs 1.42.9 (28-Dec-2013)
/dev/sdb is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=1024 blocks
65536 inodes, 262144 blocks
13107 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=268435456
8 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376

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

[root@iscis-initiator ~]# mount /dev/sdb /tmp
[root@iscis-initiator ~]# mount | tail -n 1
/dev/sdb on /tmp type ext4 (rw,relatime,seclabel,stripe=1024,data=ordered)
```

Если хотите примонтировать диск на постоянной основе, не забудьте добавить его в **/etc/fstab**. Так как это сетевое устройство, добавлять его в fstab надо с параметром **_netdev**.

Дополнительные материалы

1. Официальная документация Redhat

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. Product Documentation for Red Hat Enterprise Linux 7.
2. Основные протоколы хранения: использование и перспективы.
3. iSCSI.