OTUS

Онлайн-образование

Не забыл включить запись

Меня хорошо видно && слышно?

Ставьте плюсы, если все хорошо

Напишите в чат, если есть проблемы

Правила вебинара

- Активно участвуем
- Задаем вопросы в чат или голосом
- Off-topic обсуждаем в Slack #канал группы или #general
- Вопросы вижу в чате, могу ответить не сразу

Файловые системы и LVM

Маршрут вебинара

- LVM
- Файловые системы (ext*, xfs)
- tmpfs

Цели занятия

После занятия вы сможете:

1. Понять что такое LVM, познакомиться с терминологией
2. Познакомиться с практикой применения LVM
3. Познакомиться с основными файловыми системами
4. Узнать больше про строение файловых систем

Смысл

Зачем вам это уметь:

1. Чтобы понимать как работает LVM
2. Чтобы использовать LVM в необходимых случаях
3. Чтобы представлять структуру файловых систем и методы работы с ними

LVM

LVM

LVM (Logical Volume Manager) - специальная подсистема ядра, которая добавляет дополнительный уровень абстракции от "железа", позволяя гибко управлять дисковым пространством

LVM

LVM решает разнообразные задачи по управлению дисковой подсистемой:

- Группировка физических томов
- Создание и изменение логических томов (на лету)
- Snapshots
- Thin provisioning
- Cache volumes
- LVM mirrors
- LVM stripes

Device-mapper

Device mapper (dm) - модуль ядра, позволяющий работать с виртуальными блочными устройствами (логическими дисками). По сути посылает информацию с виртуального устройства на реальное.

Device-mapper

Некоторые из его возможностей:

- Кэширование
- Шифрование
- Зеркалирование (mirror drives)
- Multipath
- RAID (mdadm)
- Thin-provision
- Stripe
- Snapshot

Практика применения LVM

Популярные use case для LVM:

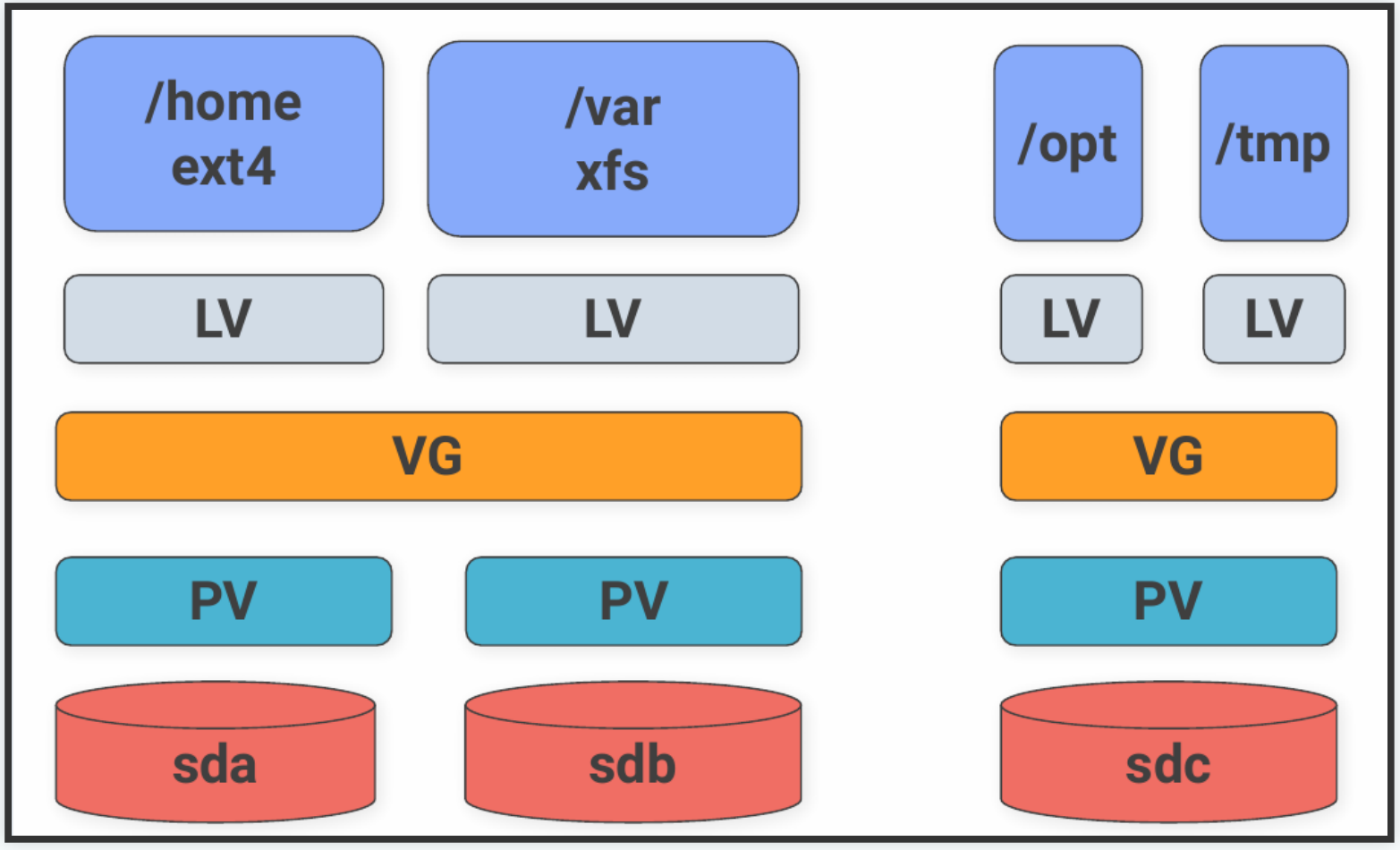
- Изменение размеров томов (в любую сторону)
- Управление большим количеством дисков (hot swap)
- Использование на десктопах
- Отказоустойчивость
- High availability кластер

Структура LVM

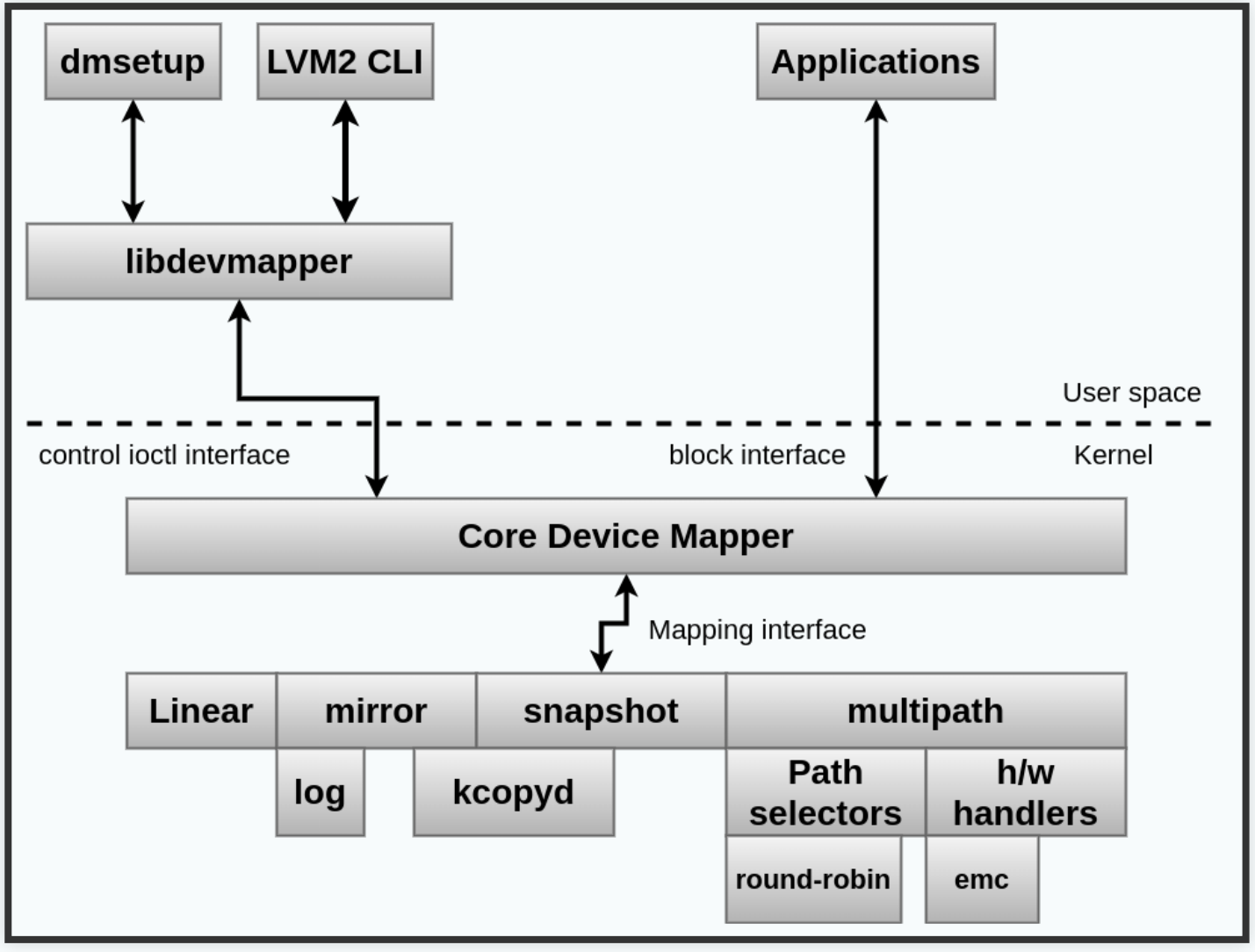
Основные элементы структуры LVM:

- PV (physical volume) - физический том, по сути любое блочное устройство
- VG (volume group) - группа физических томов
- LV (logical volume) - часть VG, доступная в виде блочного устройства

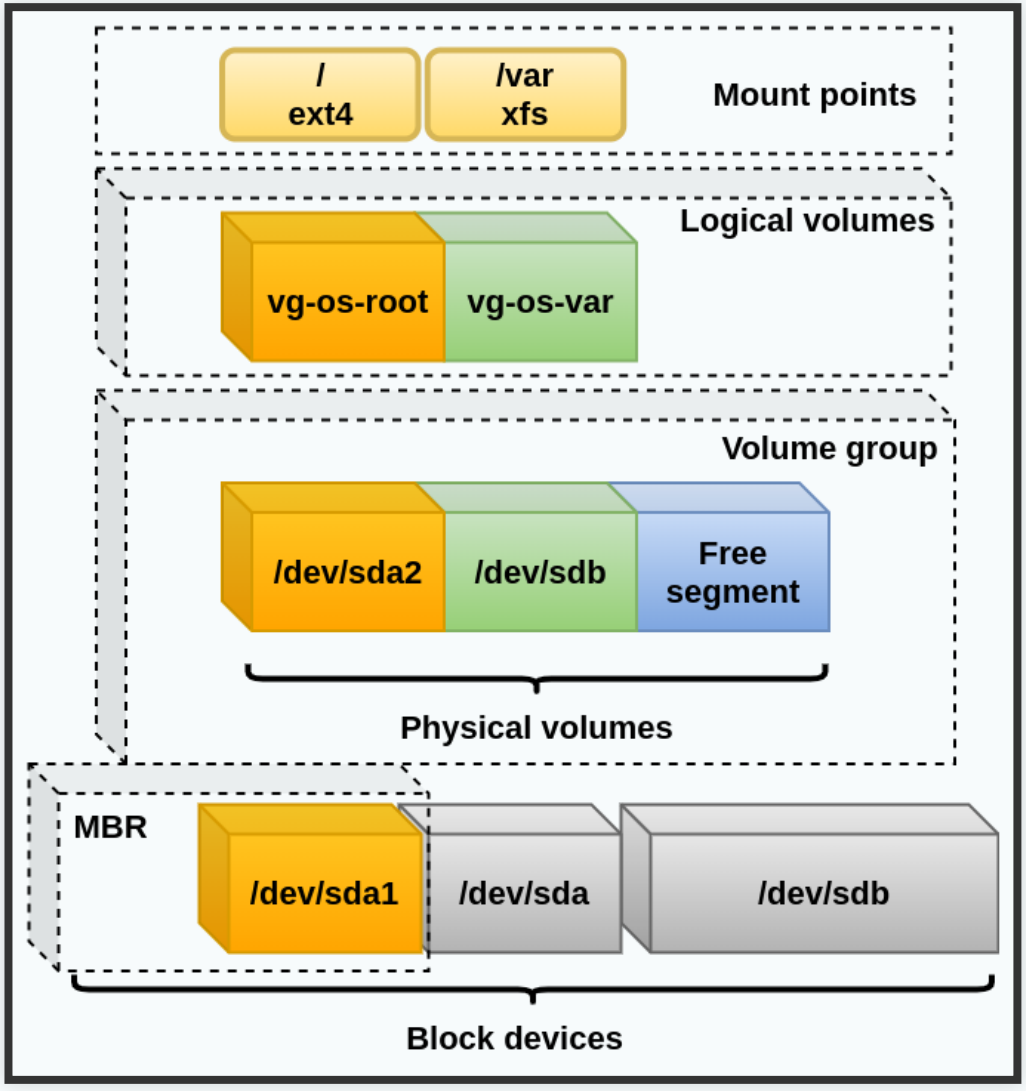
Структура LVM



Архитектура LVM



Пример использования LVM на VM



Ваши вопросы?

Управление и конфигурирование LVM

- Для управления LVM необходим пакет lvm2
- /usr/sbin/lvm - основная утилита, все остальные - лишь симлинки на нее

Создаем physical volume на блочном устройстве:

```
[root@otus ~]# pvcreate /dev/sdb
```

Создаем volume group на блочном устройстве:

```
[root@otus ~]# vgcreate vg0 /dev/sdb
```

Управление и конфигурирование LVM

Создаем дополнительный physical volume:

```
[root@otus ~]# pvcreate /dev/sdc
```

Расширяем volume group на новый physical volume:

```
[root@otus ~]# vgextend vg0 /dev/sdc
```

Вариант расширения logical volume на 10 экстентов по 4M:

```
[root@otus ~]# lvcreate -l 10 -n home vg0
```

Управление и конфигурирование LVM

Вариант расширения logical volume на 1 Гб:

```
[root@otus ~]# lvcreate -L 1G -n root vg0
```

Расширяем logical volume на 100% свободного пространства:

```
[root@otus ~]# lvcreate -l 100%FREE -n var vg0
```

Управление и конфигурирование LVM

Сканирование блочных устройств на предмет наличия LVM:

```
[root@otus ~]# vgscan
```

Сделать volume group неактивной:

```
[root@otus ~]# vgchange -ay
```

Альтернативные команды для просмотра состояний элементов LVM:

```
[root@otus ~]# pvs | vgs | lvs
```

Управление и конфигурирование LVM

Удаление элементов LVM:

```
[root@otus ~]# pvremove | vgremove | lvremove
```

Конфигурация LVM

Основные элементы конфигурации LVM:

- /etc/lvm/ - хранилище кэша конфигурации и резервных копий
- /etc/lvm/lvm.conf - основная конфигурация для утилиты lvm
- archive/ - информация за все время
- backup/ - бэкап текущей конфигурации
- cache/ - кэш
- profile/ - готовые (или самописные профили)

Конфигурация LVM

Тестирование (dry-run) восстановления конфигурации LVM из бэкапа:

```
[root@otuslinux ~]# vgcfgrestore VolGroup00 --test -f \
/etc/lvm/archive/VolGroup00_00000-2099537038.vg
```

Восстановление конфигурации LVM из бэкапа:

```
[root@otuslinux ~]# vgcfgrestore VolGroup00 -f \
/etc/lvm/archive/VolGroup00_00000-2099537038.vg
```

Далее запускаем lvscan для валидации:

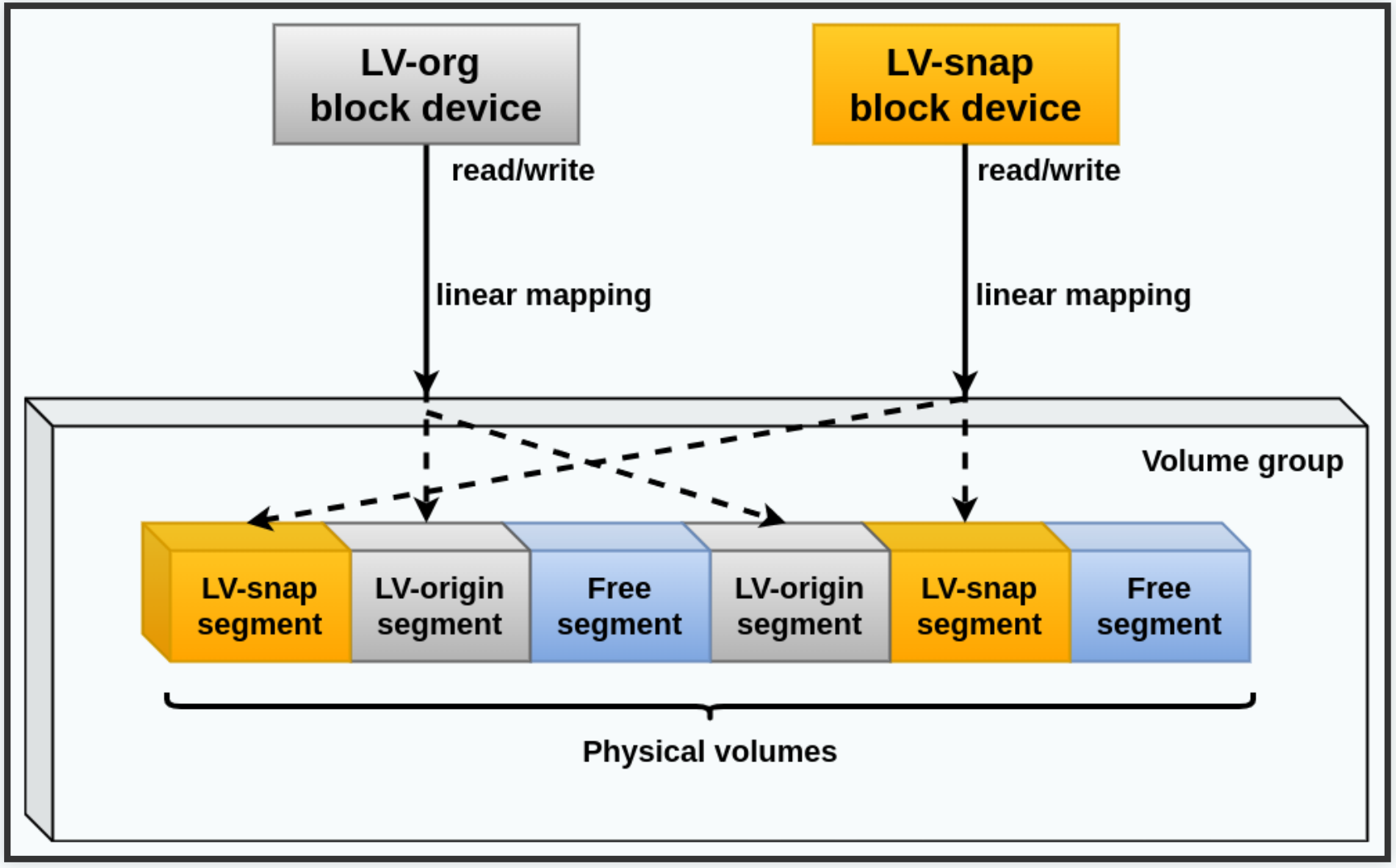
```
lvscan
```

LVM snapshots

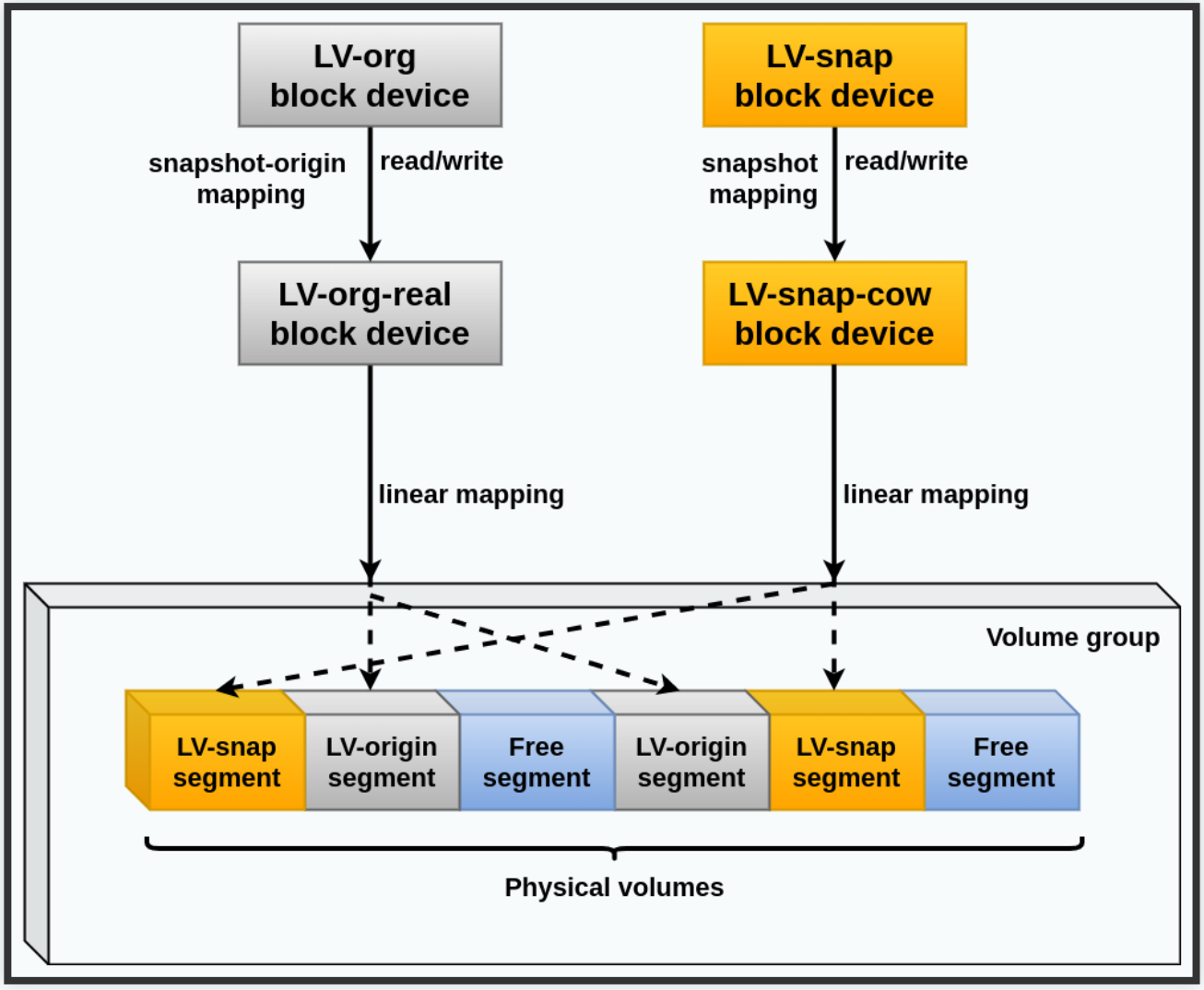
Принцип действия:

- создание нового LV
- новый LV зависит от оригинального LV
- оригинальные блоки данных копируются в новый LV перед тем, как в оригинальный LV будут записаны новые блоки данных (Copy-on-Write)

LVM snapshots



LVM snapshots



LVM snapshots

Особенности:

- необходимо наличие свободного места в VG под снапшот
- в результате при использовании снапшотов мы получаем двойную запись и как следствие - замедление дисковых операций
- удаление снапшота - быстрый процесс
- откат на снапшот - медленный процесс
- снапшот можно монтировать, в том числе в RW режиме

LVM snapshots

Создание снапшота:

```
[root@otus ~]# lvcreate -L 500M -s -n test-snap /dev/otus/test
```

Слить (смержить) снапшот с оригинальным LV:

```
[root@otus ~]# lvconvert --merge /dev/otus/test-snap
```

Удаление снапшота:

```
[root@otus ~]# lvremove /dev/otus/test-snap
```

LVM snapshots

Примерный сценарий бэкапа базы данных с помощью LVM snapshot`a:

- Сбросить данные в таблицах базы данных на диск с блокировкой таблиц на запись:

```
mysq> flush tables read lock;
```

- Создать LVM снапшот раздела, на котором расположены файлы базы:

```
lvcreate -l100%FREE -s -n mysql-backup /dev/vg0/var
```

LVM snapshots

- Разблокировать таблицы в базе данных:

```
mysq> unlock tables;
```

- Обнаружить новый снапшот:

```
lvscan
```

- Примонтировать LVM снапшота как обычную файловую систему:

```
mkdir -p /mnt/snapshot
mount /dev/vg0/mysql-backup /mnt/snapshot
```

LVM Thin Provision

Overbooking для LVM:

- возможность выделить места больше, чем есть
- используется в виртуализации и контейнеризации

Принцип:

- создается основной LV (thin pool)
- создаются зависимые LV с указанием виртуального размера

LVM Thin Provision

Основные команды:

Создаем основной LV (thin pool):

```
[root@otus ~]# lvcreate -L 100G -T vg0/lv-thinpool
```

Создаем зависимые LV:

```
[root@otus ~]# lvcreate -V 100G -T vg0/lv-thinpool -n lv1
[root@otus ~]# lvcreate -V 100G -T vg0/lv-thinpool -n lv2
[root@otus ~]# lvcreate -V 100G -T vg0/lv-thinpool -n lv3
```

LVM cache

Принцип действия кэша LVM - вынос часто используемых данных на SSD

- в основном используется на десктопах
- суть в том, что добавляется кэш LV
- в случае использования кэша нельзя использовать снапшоты

Документация по кэшу LVM:

http://man7.org/linux/man-pages/man7/lvmcache.7.html

LVM перенос данных

Перенос данных с LVM представляет собой процедуру переноса использующихся физических экстентов

Особенности:

- для перещения данных в активной системе используется команда pvmove
- pvmove разбивает данные на секции и создает временное зеркало для переноса каждой секции

LVM перенос данных

Пример

Перенос всех данных с тома /dev/sdc другим томам в группе:

```
pvmove -b /dev/sdc
```

Ваши вопросы?

Маршрут вебинара

- LVM
- Файловые системы (ext*, xfs)
- tmpfs

Файловые системы

Структура файловой системы

Файловую систему можно разделить на четыре основных компонента:

- Именованное пространство (namespace) - то как вещи (файлы, директории) представлены и организованы (иерархия)
- API - набор системных вызовов для навигации и управления объектами
- Модель безопасности - схемы для защиты, скрытия и разделения информации
- Реализация - софт для переноса логической модели на "железо"

hier[archy]

Описание иерархии файловой системы

Основная документация:

```
man hier
```

Файловая система в Linux (и в UNIX в целом) имеет иерархическую/древовидную организацию

Принято минимизировать количество разделов в корне и размещать там стартовые ветки иерархий

hier[archy]

В корне мы обычно видим следующие каталоги:

- /boot - информация необходимая для загрузки
- /bin,/sbin - системные исполняемые файлы
- /etc - файлы конфигурации системы и приложений
- /home - домашние каталоги пользователей
- /var - динамически изменяемая информация (БД, кэши, логи)
- /tmp (tmpfs) - временные файлы
- /lib[64] - системные библиотеки
- /usr - пользовательские (или системные) программы

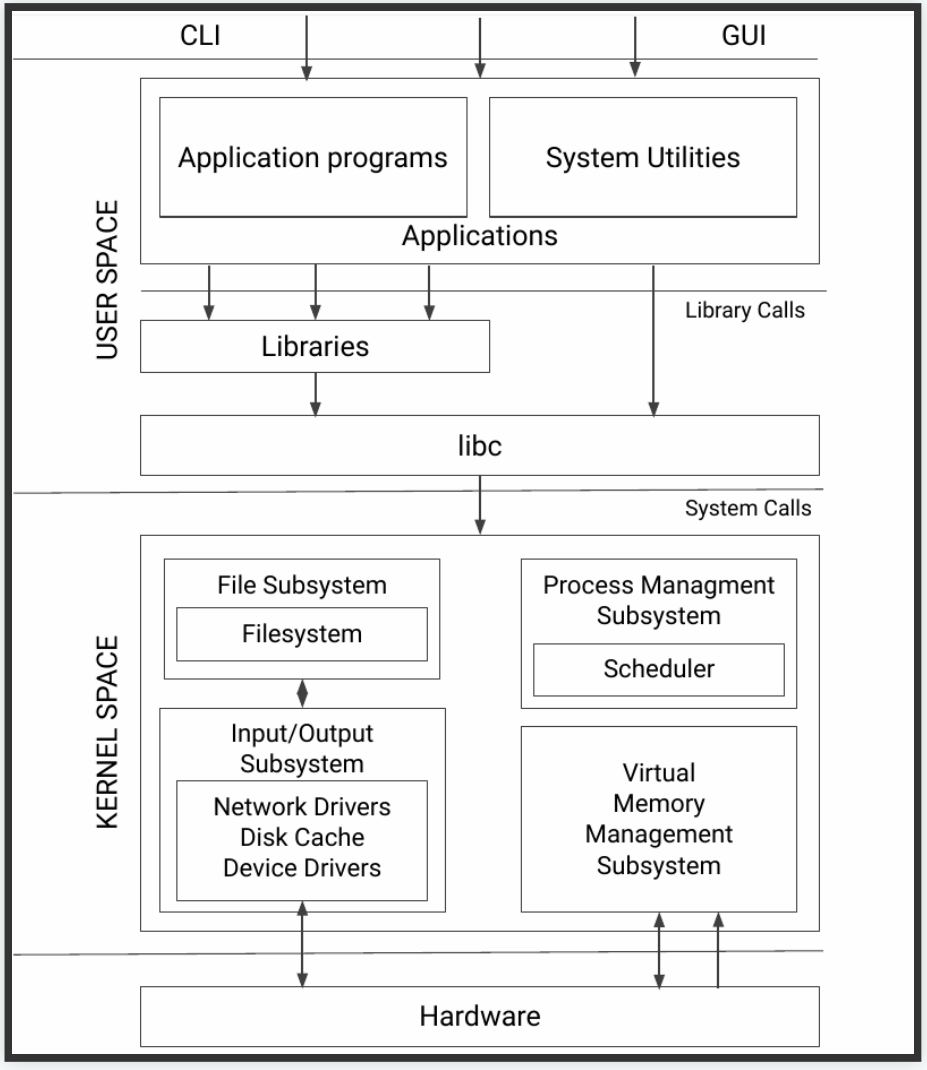
hier[archy]

В современных системах имеет смысл выделять следующие размеры разделов при установке:

- / - 8G
- /home - 8G
- /var - 16G

Для приложений стоит выделять отдельные тома (например для mysql - отдельный том в /var/lib/mysql)

Обращения к файловой системе



Псевдофайловые системы

Виртуальные файловые системы не занимают места на диске, при этом предоставляют некий интерфейс ко внутренним структурам данных в ядре

- /proc - предоставляет доступ к запущенным процессам
- /sys - предоставляет информацию об устройствах, модулях ядра, файловых системах и прочем

Псевдофайловые системы

Ссылки на документацию:

http://man7.org/linux/man-pages/man5/procfs.5.html#NOTES

http://man7.org/linux/man-pages/man5/procfs.5.html#NOTES

Страничный кэш

Страничный кэш - это кэш страниц памяти, то есть размещенные в памяти блоки данных тех файлов, к которым только производился доступ

Управляется страничный кэш утилитой **pdflush**

Страничный кэш

Параметры работы pdflush:

- **/proc/sys/vm/dirty_expire_centisecs** - время жизни dirty data в памяти
- **/proc/sys/vm/dirty_background_ratio** - объем кеша, занятого dirty data (% кеша)
- **/proc/sys/vm/dirty_ratio** - объем кеша, занятого dirty data (% общей памяти)
- **/proc/sys/vm/dirty_writeback_centisecs** - период работы pdflush

Страничный кэш

fsync() - системный вызов, записывающий все измененные данные для конкретного файла на диск (man 2 fsync)

Сброс кешей:

```
[root@otus ~]# sync
[root@otus ~]# echo 3 > /proc/sys/vm/drop_caches
```

block, superblock, inode, hardlink

Блок - минимальный адресуемый размер дискового пространства:

- Исторически равен 512 байт
- Так же это минимально выделяемый размер под файл

Суперблок - информация о файловой системе:

- размер ФС
- размер блока
- битмап занятых блоков
- расположение и размер групп блоков и таблиц inode

```
[root@otus ~]# dumpe2fs -h /dev/sda1
```

block, superblock, inode, hardlink

Inode (индексный дескриптор) - структура, содержащая информацию о файле:

- размер файла в байтах;
- идентификатор владельца файла;
- идентификатор группы-владельца файла;
- режим доступа к файлу, определяющий кто и какой доступ имеет к файлу;
- дополнительные системные и пользовательские флаги, которые дополнительно могут ограничивать доступ к файлу и его модификацию;

block, superblock, inode, hardlink

Inode (индексный дескриптор) - структура, содержащая информацию о файле:

- временные метки, отражающие время модификации индексного дескриптора (ctime, changing time), время модификации содержимого файла (mtime, modification time) и время последнего доступа к файлу (atime, access time);
- счётчик для учёта количества жёстких ссылок на файл;
- указатели на физические блоки диска, в которых хранится содержимое файла

block, superblock, inode, hardlink

Inode #2 - root

Директория - это тоже файл

- содержимое которого представляет индекс (B-tree), содержащий имя файла (ключ) и номер inode (значение)
- порядок файлов в директории не гарантируется
- перечисление всех файлов - очень медленно, но доступ к любому быстрый

Файловые системы

ext2 - исторически “стандартная” для Linux. файловая система решавшая много ограничений своих предтечей - ext и minix. Считается эталоном производительности. Поддерживается online resize

ext3 - Логическое продолжение ext2, расширены ограничения на размер файлов и тома, добавлена возможность журналирования

Файловые системы

ext4 - Логическое продолжение ext3. Номинально сильно увеличены ограничения на размер тома, по факту из коробки все еще 4Тб на том, возможность хранить ext. attributes в Inode, увеличение inode (128->256b), решен вопрос со вложенными каталогами (>32000)

Файловые системы

XFS - высокопроизводительная журналируемая файловая система родом из SGI (Silicon Graphics)

Преимущества:

- динамическая аллокация inode
- дефрагментация на лету
- потенциально лучшая производительность
- встроенные средства резервного копирования и снапшотов (xfsdump/xfsrestore)
- "отсутствие" жестких ограничений на размер файловой системы

Файловые системы

XFS - высокопроизводительная журналируемая файловая система родом из SGI (Silicon Graphics)

Недостатки:

- низкая ремонтопригодность
- выше вероятность сбоя из-за хранения большого количества данных в памяти
- невозможность уменьшения раздела с XFS

Журналирование

Журналирование - процесс записи всех изменений в журнал (лог) прежде чем они исполняются в файловой системе

Журнал - хранит лог изменения метаданных

Процесс восстановления файловой системы после сбоя - по факту применение журнала вместо полной проверки с помощью fsck

Журналирование

Для ext4 есть три варианта (mount -o data=)

- **journal** - пишем сначала в журнал, самый медленный метод
- **ordered** - пишем сначала файловую систему, потом журнал
- **writeback** - не гарантируется порядок изменений, в файле после сбоя может появиться блок данных, который был до сбоя удалён; самый быстрый метод

Ваши вопросы?

Знакомство с fstab

/etc/fstab - конфигурационный файл, содержащий информацию о блочных устройствах, файловых системах на них и о том, как они будут интегрированы в систему

Пример файла /etc/fstab:

```
[root@otus ~]# grep -v '^#' /etc/fstab

/dev/mapper/os--root-root /                       ext4    default
UUID=fc909a8b-a9e6-4e80-8034-77662ec0b96e /boot
```

Знакомство с fstab

Особенности:

- предпочтительней указывать UUID, а не имя блочного устройства, так как имена могут меняться
- UUID можно узнать командой blkid
- UUID при необходимости тоже можно менять

Опции монтирования файловой системы

Список опций можно посмотреть в мане:

```
man mount
```

Распространенные опции монтирования:

- noatime и nodiratime - выключает запись атрибутов atime и diratime в файлы и каталоги
- strictatime - включает запись atime и diratime
- data - задает параметры работы с журналом

Опции монтирования файловой системы

- user - задается пользователь от имени которого будет монтироваться ресурс
- noexec - запрет на выполнение скриптов в примонтированной файловой системе
- noauto - альтернатива параметру auto, ресурс не будет монтироваться автоматически

Опции монтирования файловой системы

Способы просмотра процессов, обращающихся к файловой системе:

- утилита fuser выводит идентификаторы всех процессов, обращающихся к файлам или каталогам указанной файловой системы

```
[root@otus ~]# fuser -c /usr
```

- утилита lsof аналогична по функционалу, но выдает больше информации

```
[root@otus ~]# lsof /usr
```

Немного про fsck

Утилита fsck используется для восстановления файловых систем в непредсказуемом состоянии

- при загрузке происходит проверка файловых систем из /etc/fstab
- перед проверкой вручную файловую систему обязательно нужно отмонтировать!
- журналируемые файловые системы проверяются быстрее за счет того, что fsck проверяет только последние записанные данные

Немного про fsck

Пример использования fsck:

```
[root@otus ~]# umount /mnt
[root@otus ~]# fsck -y /dev/sdb
```

Ваши вопросы?

Маршрут вебинара

- LVM
- Файловые системы (ext*, xfs)
- tmpfs

tmpfs

tmpfs. ramfs. rootfs. initramfs

Файловые системы tmpfs, ramfs, rootfs, initramfs - это методы предоставления доступа к кэширующему механизму на уровне пользователя

Документация по ramfs, rootfs and initramfs:

https://www.kernel.org/doc/Documentation/filesystems/rootfs-initramfs.txt

tmpfs. ramfs. rootfs. initramfs

Пример монтирования tmpfs:

```
[root@otus ~]# # mount -t tmpfs none /mnt -o size=100M
```

swap

swap - это процесс когда страницы памяти копируются на заранее размеченное место на диске

Делается это затем, чтобы освободить место в памяти

Просмотр текущего состояния swap на всех разделах:

```
[root@otus ~]# swapon -s
```

swap

Просмотр состояния оперативной памяти:

```
[root@otus ~]# free -h
```

Создание размеченной области под swap:

```
[root@otus ~]# mkswap /dev/sdb
```

Включение/выключение swap на указанном разделе:

```
[root@otus ~]# swapon[off] /dev/sdb
```

Полезные утилиты

Некоторые полезные утилиты:

```
[root@otus ~]# df -Th
[root@otus ~]# du
[root@otus ~]# stat
[root@otus ~]# ncdu
[root@otus ~]# lsof
[root@otus ~]# fuser
[root@otus ~]# fsck
[root@otus ~]# mkfs.*
[root@otus ~]# mount | column -t
```

Рефлексия

- Назовите пожалуйста 3 момента, которые вам запомнились в процессе занятия
- Что вы будете применять в работе из сегодняшнего вебинара?

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Приходите на следующие вебинары

Спасибо за внимание!