Настраиваем использование оперативной памяти при работе с ZFS в Proxmox VE

ZFS - продвинутая файловая система с огромным количеством возможностей, что, безусловно, привлекает к ней администраторов. Тем более, что в Proxmox VE использовать ZFS действительно просто и для этого вам не потребуются какие-то особые знания. С одной стороны это хорошо - нажал на кнопку и получил результат, с другой - может сыграть злую шутку. Основная проблема использования ZFS в Proxmox - высокое потребление оперативной памяти. Сегодня мы рассмотрим причины этого явления и дадим советы по оптимизации.

Для начала коснемся некоторых распространенных заблуждений. Бытует мнение, что оперативная память сейчас неприлично дешева и доступна практически каждому. Да, это так, если смотреть на ее цену в прайс-листах, но на самом деле оперативная память - **дорогой ресурс**. Дело в том, что максимальное количество памяти для системы - конечное значение и бесконечно увеличивать его, в отличие от дискового пространства мы не можем.

А в системах виртуализации память расходуется очень быстро, поэтому ZFS способна серьезно ухудшить ситуацию и даже стать причиной нестабильной работы гипервизора.

Еще один распространенный миф - это использование ZFS с памятью без коррекции ошибок ECC. Согласно каноническому тексту это чревато полным разрушением структуры ZFS с потерей данных. На самом деле это не так, дадим слово одному из сооснователей ZFS и ее текущему разработчику Мэтью Аренсу:

> В ZFS нет ничего особенного, что требует / поощряет использование ECC памяти больше, чем любая другая файловая система.

Поэтому если вы хотите использовать ZFS вместе с обычным железом, без поддержки ECC - используйте. Все чем вы рискуете - это повреждением данных при записи из памяти на диск, но эти риски будут не выше, чем если бы вы использовали другую файловую систему.

Перейдем теперь к сути проблемы. Добавив в Proxmox VE хранилище с использованием ZFS администратор сразу же столкнется с резким ростом потребления оперативной памяти. Это происходит потому, что ZFS использует для кеширования до 50% установленной оперативной памяти хоста, что серьезно снижает количество доступной памяти для виртуальных машин.

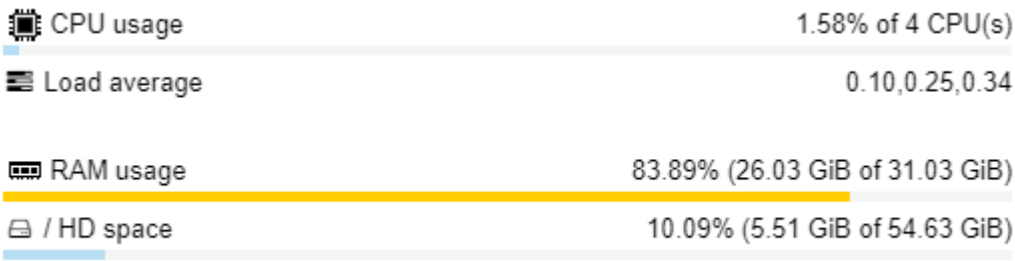
Допустим у вас на хосте установлено 64 ГБ ОЗУ, задействовав ZFS у вас останется только 32 ГБ, вычтем отсюда собственное потребление гипервизора и у нас останется всего около 30 ГБ памяти для виртуальных машин. Чем это чревато? Если памяти для всех достаточно, то узел будет продолжать работать, только вот его емкость серьезно уменьшится, развернуть новые виртуальные машины вы уже не сможете.

Если же памяти не хватает, то все может быть значительно хуже. Сначала система уйдет в своп, поэтому важно, чтобы раздел или файл подкачки располагался на быстром носителе - SSD или NVMe, но даже в этом случае просадка производительности неизбежна.

Важно! Не располагайте пространство подкачки на ZFS, это может привести к полной блокировке системы в условиях недостатка памяти.

В самом плохом варианте, при критическом дефиците памяти сработает **OOM Killer** и одна из ваших виртуальных машин будет принудительно остановлена.

При этом многие в оценке потребления памяти ориентируются на показания веб-интерфейса Proxmox VE, который не совсем верно отражает действительность. Например, глядя на такое можно подумать, что определенный запас по памяти есть.



На самом деле свободная часть оперативной памяти редко бывает свободна, обычно там располагается кеш VFS и его сброс может негативно сказаться на производительности, поэтому система может принять решение отправить работающий процесс в подкачку или вообще "пристрелить" его через OOM Killer, лишь бы не сбрасывать кеш VFS.

Реальное состояние дел с распределением памяти вы можете посмотреть командой:

```
free -h
```

В общем, проблема понятна. Какие могут быть решения? Начнем с оптимизации использования памяти виртуальными машинами.

Настройка KSM

KSM (Kernel Same-page Merging) - специальная технология ядра Linux, позволяющая объединять страницы памяти разных процессов, тем самым добиваясь ее экономии, фактически данный процесс можно назвать дедупликацией оперативной памяти. При наличии большого количества однотипных виртуальных машин эффект от использования KSM может быть достаточно ощутимым.

Обратите внимание! KSM работает только с виртуальными машинами KVM, для контейнеров LXC эта технология не применима.

KSM сразу включен Proxmox VE, но его настройки могут быть недостаточно оптимальными. Поэтому откроем файл **/etc/ksmtuned.conf** и ознакомимся с его содержимым. Все опции в нем закомментированы и содержат значения по умолчанию, если мы хотим изменить значение, то строку надо раскомментировать.

Начнем с двух опций:

```
KSM_THRES_COEF=20
KSM_THRES_CONST=2048
```

KSM_THRES_COEF - основной параметр, указывающий при каком объеме оставшейся свободной памяти, начнет работать **KSM**. Несложно подсчитать, что для системы с 64 ГБ ОЗУ это произойдет после того, как будут заняты 51,2 ГБ памяти. Второй параметр **KSM_THRES_CONST** указывает аналогичное значение в МБ, вы можете использовать его для более точного задания порогов, при различных значениях параметров сработает самый низкий порог.

Многим может показаться, что KSM начинает работать слишком поздно, но этому тоже есть своя причина - операция по сравнению и объединению страниц памяти довольно затратна по использованию процессора, поэтому указанные значения - это некоторый разумный компромисс. Если же у вас нет недостатка в процессорных ресурсах, но актуально рациональное использование памяти - можете попробовать изменить эти параметры.

Следующие два параметра в целом отвечают за нагрузку на CPU:

```
KSM_MONITOR_INTERVAL=60
KSM_SLEEP_MSEC=100
```

KSM_MONITOR_INTERVAL - задает промежуток между сканированиями в секундах. Сканирование происходит блоками по 16 ГБ, пауза между этими процессами задается опцией **KSM_SLEEP_MSEC.** Увеличение этих параметров снижают нагрузку на процессор, но уменьшают эффективность работы KSM.

Минимальное и максимальное количество страниц, которые будут просканированы и объединены задается параметрами:

```
KSM_NPAGES_MIN=64
KSM_NPAGES_MAX=1250
```

Без особой нужды и понимания того, что вы делаете, редактировать их не следует.

Агрессивность работы KSM задается следующими параметрами:

```
KSM_NPAGES_BOOST=300
KSM_NPAGES_DECAY=-50
```

KSM_NPAGES_BOOST определяет какое количество сканируемых страниц будет прибавлено к текущему значению при превышении порога оставшейся свободной памяти, **KSM_NPAGES_DECAY**, наоборот, уменьшает количество страниц, когда свободной памяти окажется больше, чем установлено в пороговых значениях. Таким образом достигается баланс между количеством сканируемых и объединяемых страниц и необходимым запасом свободной памяти.

Но приведенные значения совсем не догма и если у вас большое количество однотипных виртуальных машин, то можете поэкспериментировать с настройками KSM, начав с изменения KSM_THRES_COEF.

После внесения изменений не забудьте перезапустить службу командой:

```
systemctl restart ksmtuned
```

При выборе значений будьте благоразумны, помните, что за уплотнение памяти вы платите повышенной нагрузкой на процессор.

Настройка ARC

ARC (*Adaptive Replacement Cache, кэш адаптивной замены*) - собственный эффективный механизм кеширования ZFS использующий достаточно сложные и эффективные алгоритмы. Основной смысл его работы - максимально исключить обращения к диску на чтение в пользу использования кешированных данных. И чем эффективнее работает ARC, чем больше попаданий в кеш, тем тяжелее становятся его элементы и тем труднее их из памяти вытеснить. В результате ZFS практически всегда будет занимать под ARC все доступное ей пространство. С одной стороны, это хорошо, но не будем забывать, что память - дорогой ресурс, а соверменные хранилища могут состоять из быстрых SSD или NVMe накопителей.

В большинстве случаев имеет смысл радикально ограничить ZFS в потреблении памяти, высвободив ее для виртуальных машин.

Классическая формула расчета предполагает:

```
1 ГБ для хоста + 1 ГБ на 1 ТБ хранилища
```

При этом не рекомендуется использовать значения кеша менее 3 ГБ.

Современные рекомендации предполагают большие размеры выделяемой памяти:

```
1 ГБ + 4-5 ГБ на 1 ТБ хранилища
```

А если вы используете дедупликацию, то:

```
1 ГБ + 5-6 ГБ на 1 ТБ хранилища
```

Исходя из этого и рекомендаций по минимальному размеру можно считать достаточно оптимальным значением для небольших хранилищ размер кеша 4 - 8 ГБ. Размер кеша ZFS задается в байтах и вычислить его можно по формуле:

```
N*1048576*1024
```

Где N - нужное значение в ГБ.

Чтобы добавить собственные настройки лимитов ARC создайте, если вы не сделали этого ранее, файл **/etc/modprobe.d/zfs.conf** и откройте его для редактирования. Все это можно сделать одной командой:

```
nano /etc/modprobe.d/zfs.conf
```

Если вы предпочитаете редактор mc, замените nano на mcedit. Если файл не существует, то он будет создан и открыт для редактирования, если существует - просто открыт для редактирования.

Теперь внесем в него следующие строки, задающие минимальный и максимальный размер кеша:

```
options zfs zfs_arc_min=4294967296
options zfs zfs_arc_max=8589934592
```

Затем обновим образ начальной файловой системы:

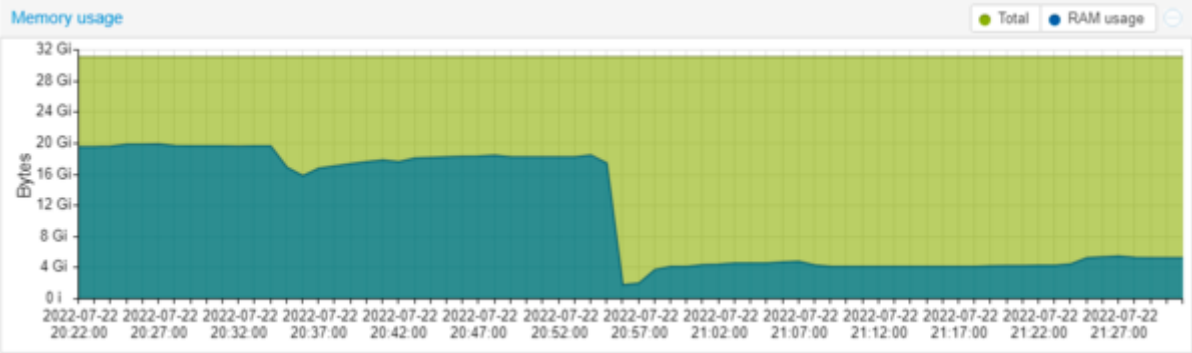
```
update-initramfs -u
```

Если ваша система использует UEFI, то дополнительно выполните:

```
pve-efiboot-tool refresh
```

После чего выполните перезагрузку.

Результат обычно виден сразу и, что называется, невооруженным глазом.



Как видим, даже минимальное понимание происходящих процессов и грамотный подход к распределению оперативной памяти дает отличный результат, позволяя использовать все возможности ZFS, не жертвуя столь необходимой для виртуальных машин памятью.