Контроль расходования ресурсов

StarVault — это система, управляемая API, поэтому все коммуникации между клиентами и StarVault осуществляются через StarVault API.

По мере увеличения количества клиентских приложений неавторизованные приложения могут снижать производительность StarVault. Проблема часто возникает из-за приложений, которые:

- Генерирование неограниченного количества аренд и/или загрузка слишком большого объема данных в StarVault, что приводит к исчерпанию доступной памяти бэкенда хранилища
- Потребление ошибочно большого количества пропускной способности, что приводит к внутреннему тротлингу кластера StarVault в целом

1. Решение

Установите квоты на ресурсы StarVault, чтобы защитить стабильность среды StarVault и сети, а также потребление ресурсов хранения от неконтролируемого поведения приложений и распределенных атак типа "отказ в обслуживании" (DDoS).

Операторы StarVault могут управлять тем, как приложения запрашивают ресурсы StarVault, а также хранилищем и сетевой инфраструктурой StarVault, задавая следующие параметры:

Характеристика	**Описание**
Квоты на ограничение скорости	Ограничьте максимальное количество запросов в секунду (RPS) к системе или монтированию, чтобы защитить пропускную способность сети.

Чтобы установить квоты на ограничение скорости и квоты на количество арендованных ресурсов, используйте конечную точку `sys/quotas/<type>`. Каждая квота ресурса имеет имя для идентификации правила квоты. Для управления отдельным правилом квоты конечной точкой становится `sys/quotas/<type>/<name>`.

`<type>` может быть:

- `rate-limit`: квота предельной скорости

2. Необходимое условие

Для выполнения задач, описанных в этом руководстве, вам потребуется StarVault. Для установки StarVault обратитесь к руководству по установке StarVault. Убедитесь, что сервер StarVault инициализирован и распечатан.

3. Лабораторная установка

1. Откройте терминал и запустите dev-сервер StarVault с `root` в качестве корневого токена.

   ```bash
   starvault server -dev -dev-root-token-id root
   ```

   По умолчанию dev-сервер StarVault работает по адресу `127.0.0.1:8200`. Сервер также инициализируется и распечатывается.

   > Не запускайте dev-сервер StarVault в боевой среде. Этот подход используется здесь только для упрощения процесса распечатывания в качестве демонстрации.

2. Экспортируйте переменную окружения для `starvault` CLI, чтобы обратиться к серверу StarVault.

   ```bash
   export STARVAULT_ADDR=http://127.0.0.1:8200
   ```

3. Экспортируйте переменную окружения для `starvault` CLI, чтобы аутентифицироваться на сервере StarVault.

   ```bash
   export STARVAULT_TOKEN=root
   ```

Сервер StarVault готов.

4. Настройка квот ресурсов

Чтобы настроить квоту ресурсов, используйте конечную точку `sys/quotas/config`.

Параметр	Тип	Описание
`enable_rate_limit_audit_logging`	boolean	Включите или отключите ведение логов аудита, когда запросы отклоняются из-за нарушения квоты лимита скорости. По умолчанию ведение логов аудита отключено (`false`).

По умолчанию запросы, отклоненные из-за нарушения квоты ограничения скорости, не записываются в логи аудита. Поэтому, если хотите регистрировать отклоненные запросы для отслеживания, то установие параметру `enable_rate_limit_audit_logging` значение `true`. Запросы, отклоненные по причине достижения квоты на количество арендованных мест, всегда записываются в логах, и вам не нужно устанавливать какой-либо параметр.

Включение регистрации аудита ограничения скорости может повлиять на производительность StarVault, если объем отклоненных запросов велик.

4.1. Комнды cli

1. Включите устройство файлового аудита, которое выводит данные в файл `/var/log/starvault-audit.log` (или в нужное вам место).

```bash
starvault audit enable file file_path="/var/log/starvault-audit.log"
```

2. Чтобы включить логирование аудита для квот на ограничение скорости, выполните следующую команду.

```bash
starvault write sys/quotas/config enable_rate_limit_audit_logging=true
```

3. Прочитайте конфигурацию квоты для проверки.

```bash
starvault read sys/quotas/config

Key                                 Value
---                                 -----
enable_rate_limit_audit_logging     true
enable_rate_limit_response_headers  false
rate_limit_exempt_paths             []
```

4.2. Вызов API с помощью cURL

1. Включите устройство аудита файлов.

   Сначала создайте полезную нагрузку HTTP-запроса, указав путь к логам аудита `/var/log/starvault-audit.log` (или желаемое местоположение файла).

   ```bash
   tee audit-payload.json <<EOF
   {
     "type": "file",
     "options": {
         "file_path": "/var/log/starvault-audit.log"
     }
   }
   EOF
   ```

   Используйте конечную точку `sys/audit`, чтобы включить журнал аудита файлов.

   ```bash
   curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
       --request POST \
       --data @audit-payload.json \
       $STARVAULT_ADDR/v1/sys/audit/file
   ```

   Установите целевой путь файла (`file_path`) в нужное вам место.

2. Создайте полезную нагрузку HTTP-запроса, чтобы включить логирование аудита квоты ограничения скорости.

   ```bash
   tee payload.json <<EOF
   {
      "enable_rate_limit_audit_logging": true
   }
   EOF
   ```

3. Вызовите конечную точку `sys/quotas/config`.

   ```bash
   curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
       --request POST \
       --data @payload.json \
       $STARVAULT_ADDR/v1/sys/quotas/config
   ```

4. Прочитайте конфигурацию квоты для проверки.

   ```bash
   curl -s --header "X-Vault-Token: $STARVAULT_TOKEN" \
       $STARVAULT_ADDR/v1/sys/quotas/config | jq -r ".data"
   ```

   Пример вывода:

```
TERMINAL
{
  "enable_rate_limit_audit_logging": true,
  "enable_rate_limit_response_headers": false,
  "rate_limit_exempt_paths": []
}
```

5. Квоты на ограничение скорости

Квоты на ограничение скорости предназначены для защиты StarVault от внешних распределенных атак типа "отказ в обслуживании" (DDoS) и являются основой модели безопасности StarVault.

5.1. Параметры

Параметр	Тип	Описание
`name`	string	Имя правила для квоты
`path`	string	Целевой путь или пространство имен для применения правила квоты. Пустой путь настраивает глобальную квоту ограничения скорости
`rate`	float	Скорость количества разрешенных запросов в секунду (RPS)
`role`	string	Роль входа в систему, к которой следует применить эту квоту. Когда этот параметр установлен, путь должен быть настроен на действительный метод аутентификации с концепцией ролей.
`interval`	second	Продолжительность, в течение которой будет действовать ограничение скорости (по умолчанию 1 секунда).
`block_interval`	string	Если установлено, то когда клиент достигает порога ограничения скорости, ему запрещается выполнять дальнейшие запросы до истечения `block_interval` .

Путь может быть полностью определенным и заканчиваться символом **(например,** `auth/token/create`).

5.1.1. Комнды cli

1. Создайте квоту с именем `global-rate`, которая ограничивает входящую нагрузку до 1500 запросов в секунду.

```
starvault write sys/quotas/rate-limit/global-rate rate=1500
```

2. Прочитайте правило `global-rate`, чтобы проверить его конфигурацию.

```
$ starvault read sys/quotas/rate-limit/global-rate

Key               Value
---               -----
block_interval    0
inheritable       true
interval          1
name              global-rate
path              n/a
rate              1500
role              n/a
type              rate-limit
```

Если отсутствует `path`, то правило квоты применяется к глобальному уровню, а не к конкретному монтированию.

3. Создайте квоту ограничения скорости с именем `transit-limit`, которая ограничивает доступ к механизму секретов Transit до 1000 запросов в минуту (60 секунд).

   Во-первых, включите механизм секретов Transit в `transit`.

```
$ starvault secrets enable transit
Success! Enabled the pki secrets engine at: transit/
```

   Во-вторых, создайте квоту на ограничение скорости `transit-limit`.

```
starvault write sys/quotas/rate-limit/transit-limit \
   path="transit" \
   rate=1000 \
   interval=60
```

   Вывод:

```
Success! Data written to: sys/quotas/rate-limit/transit-limit
```

4. Прочитайте правило `transit-limit`, чтобы проверить его конфигурацию.

```
starvault read sys/quotas/rate-limit/transit-limit
```

Пример вывода:

```
Key               Value
---               -----
block_interval    0
inheritable       true
interval          60
name              transit-limit
path              transit/
rate              1000
role              n/a
type              rate-limit
```

Ограничение скорости установлено на 1000 запросов в 60 секунд.

5.1.1.1. Детализация пути

Вы можете задать путь (`path`) глубже, чем точка монтирования (в данном примере `transit/`).

1. Создайте квоту ограничения скорости с именем `transit-order`, чтобы ограничить количество запросов на шифрование данных с использованием ключа `orders` до 500 в секунду.

   Во-первых, включите создайте ключ шифрования с именем, `orders`.

```
$ starvault write -f transit/keys/orders

Key                       Value
---                       -----
allow_plaintext_backup    false
auto_rotate_period        0s
deletion_allowed          false
derived                   false
exportable                false
imported_key              false
keys                      map[1:1695147293]
latest_version            1
min_available_version     0
min_decryption_version    1
min_encryption_version    0
name                      orders
```

```
supports_decryption       true
supports_derivation       true
supports_encryption       true
supports_signing          false
type                      aes256-gcm96
```

Во-вторых, создайте квоту на ограничение скорости `transit-order`.

BASH

```
starvault write sys/quotas/rate-limit/transit-order \
   path="transit/encrypt/orders" \
   rate=500
```

Вывод:

TERMINAL

```
Success! Data written to: sys/quotas/rate-limit/transit-order
```

2. Проверьте конфигурацию квоты ограничения скорости.

BASH

```
starvault read sys/quotas/rate-limit/transit-order
```

Вывод:

TERMINAL

```
Key               Value
---               -----
block_interval    0
inheritable       true
interval          1
name              transit-order
path              transit/encrypt/orders
rate              500
role              n/a
type              rate-limit
```

5.1.2. Вызов API с помощью cURL

Создайте квоту с именем `global-rate`, которая ограничивает входящую нагрузку до 1500 запросов в секунду.

1. Вызовите конечную точку `sys/quotas/rate-limit`.

BASH

```
curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
    --request POST \
    --data '{ "rate": 1500 }' \
    $STARVAULT_ADDR/v1/sys/quotas/rate-limit/global-rate
```

2. Прочитайте только что созданное правило квоты `global-rate`.

```
curl -s --header "X-Vault-Token: $STARVAULT_TOKEN" \
    $STARVAULT_ADDR/v1/sys/quotas/rate-limit/global-rate | jq -r ".data"
```

Вывод:

```
{
   "block_interval": 0,
   "interval": 1,
   "name": "global-rate",
   "path": "",
   "rate": 1500,
   "role": "",
   "type": "rate-limit"
}
```

Если отсутствует `path`, то правило квоты применяется к глобальному уровню, а не к конкретному монтированию.

3. Создайте квоту ограничения скорости с именем `transit-limit`, которая ограничивает доступ к механизму секретов Transit до 1000 запросов в минуту (60 секунд).

   Во-первых, включите механизм секретов Transit в `transit`.

   ```
   curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
       --request POST \
       --data '{"type":"transit"}' \
       $STARVAULT_ADDR/v1/sys/mounts/transit
   ```

   Во-вторых, создайте квоту на ограничение скорости `transit-limit`.

   ```
   curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
      --request POST \
      --data '{"path": "transit", "rate": 1000, "interval": 60 }' \
      $STARVAULT_ADDR/v1/sys/quotas/rate-limit/transit-limit
   ```

4. Прочитайте правило `transit-limit`, чтобы проверить его конфигурацию.

   ```
   curl -s --header "X-Vault-Token: $STARVAULT_TOKEN" \
      $STARVAULT_ADDR/v1/sys/quotas/rate-limit/transit-limit | jq -r ".data"
   ```

Пример вывода:

```
{
   "block_interval": 0,
   "interval": 60,
   "name": "transit-limit",
```

```
    "path": "transit/",
    "rate": 1000,
    "role": "",
    "type": "rate-limit"
}
```

Ограничение скорости установлено на 1000 запросов в 60 секунд.

5.1.2.1. Детализация пути

Вы можете задать путь (`path`) глубже, чем точка монтирования (в данном примере `transit/`).

1. Создайте квоту ограничения скорости с именем `transit-order` , чтобы ограничить количество запросов на шифрование данных с использованием ключа `orders` до 500 в секунду.

   Во-первых, включите создайте ключ шифрования с именем, `orders` .

   BASH
   ```
   curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
       --request POST \
       $STARVAULT_ADDR/v1/transit/keys/orders
   ```

   Во-вторых, создайте квоту на ограничение скорости `transit-order` .

   BASH
   ```
   curl --header "X-Vault-Token: $STARVAULT_TOKEN" \
      --request POST \
      --data '{ "path": "transit/encrypt/orders", "rate": 500 }' \
      $STARVAULT_ADDR/v1/sys/quotas/rate-limit/transit-order
   ```

2. Проверьте конфигурацию квоты ограничения скорости.

   BASH
   ```
   curl -s --header "X-Vault-Token: $STARVAULT_TOKEN" \
      $STARVAULT_ADDR/v1/sys/quotas/rate-limit/transit-order | jq -r ".data"
   ```

Вывод:

TERMINAL
```
{
   "block_interval": 0,
   "interval": 1,
   "name": "transit-order",
   "path": "transit/encrypt/orders",
   "rate": 500,
   "role": "",
   "type": "rate-limit"
}
```

Следующий шаг
Обязательно посетите раздел "Тест на понимание квот ресурсов", чтобы узнать, как работает применение квот ресурсов.

6. Тест на понимание квот на ресурсы

После того, когда вы изучили основные команды, проверьте, как они себя ведут.

6.1. Тест на определение квоты на ограничение скорости

1. Включите механизм транзита секретов, если он не включен.

```
starvault secrets enable transit
```

2. Удалите определение квоты, которое было создано ранее

```
starvault delete sys/quotas/rate-limit/transit-limit
```

3. Создайте ключ шифрования "test".

```
starvault write -f transit/keys/test
```

4. Для демонстрации создайте квоту ограничения скорости `test-transit` так, чтобы вы могли делать только 1 запрос в минуту.

```
starvault write sys/quotas/rate-limit/rate-test \
   path=transit \
   rate=1 \
   interval=1m
```

5. Создайте скрипт `test-encryption.sh`, который выполняет запрос на шифрование данных с помощью ключа `test`.

```
tee test-encryption.sh <<EOF
echo "Request 1"
starvault write transit/encrypt/test plaintext=$(base64 <<< "4111 1111 1111
1111")
echo "\nRequest 2"
starvault write transit/encrypt/test plaintext=$(base64 <<< "4222 2222 2222
2222")
echo "\nRequest 3"
starvault write transit/encrypt/test plaintext=$(base64 <<< "4333 3333 3333
```

```
3333")
EOF
```

6. Убедитесь, что сценарий является исполняемым.

```bash
chmod +x test-encryption.sh
```

7. Запустите скрипт, чтобы посмотреть, как поведет себя правило квоты.

```bash
./test-encryption.sh
```

Вывод:

```terminal
Request 1
Key            Value
---            -----
ciphertext
vault:v1:6zY2JVtuw3r11ST/JjMkmoNbbcCpQwGbJJ0BJfByk//GLQof8mcxAoerq5Y++BZK
key_version    1

Request 2
Error writing data to transit/encrypt/test: Error making API request.

URL: PUT http://127.0.0.1:8200/v1/transit/encrypt/test
Code: 429. Errors:

* request path "transit/encrypt/test": rate limit quota exceeded

Request 3
Error writing data to transit/encrypt/test: Error making API request.

URL: PUT http://127.0.0.1:8200/v1/transit/encrypt/test
Code: 429. Errors:

* request path "transit/encrypt/test": rate limit quota exceeded
```

Второй и третий запросы не прошли из-за лимита.

8. Ранее вы настроили квоты на ресурсы, чтобы включить ведение логов аудита запросов, отклоненных из-за нарушения правила квоты лимита скорости. Проверьте журнал аудита на наличие такой записи.

```bash
more /var/log/starvault-audit.log | jq
```

Если путь к журналу аудита не является `/var/log/starvault-audit.log`, обязательно установите правильный путь.

TERMINAL

```
...snip...
   "request": {
      "id": "4170a181-6a18-589b-f026-a62ad08ae644",
      "operation": "update",
      "namespace": {
         "id": "root"
      },
      "path": "transit/encrypt/test",
      "data": {
         "plaintext": "hmac-
sha256:50978c7cb3f0f463491a26152353f485b579d1e8759ec7b7b7c28f3c5b9232cd"
      },
      "remote_address": "127.0.0.1",
      "remote_port": 61028
   },
   "error": "request path \"transit/encrypt/test\": rate limit quota
exceeded"
}
```

Вы должны найти сообщение об ошибке, указывающее на превышение квоты лимита скорости. Вы можете проследить логи аудита, чтобы узнать количество запросов, которые были отклонены из-за превышения квоты. Возможно, все работает как надо, а возможно, вы обнаружите подозрительные действия в отношении определенного пути.

9. Создайте еще одну квоту ограничения скорости, чтобы указать путь к `transit/encrypt/test`, где разрешенная скорость равна 2 с интервалом в 1 минуту.

BASH

```
starvault write sys/quotas/rate-limit/encryption-limit \
   path="transit/encrypt/test" \
   rate=2 \
   interval=1m
```

10. Запустите скрипт еще раз, чтобы посмотреть, как поведет себя правило квоты.

BASH

```
./test-encryption.sh
```

Вывод:

TERMINAL

```
Request 1
Key            Value
---            -----
ciphertext
vault:v1:44sOnzePg8c0XaxzGeGzwJ94UZ1uuif8p0nGuze8utbvWVhqNHDiIwo8VZmPrcli
key_version    1

Request 2
Key            Value
---            -----
```

```
ciphertext
vault:v1:w9tyXqoXH6uXuOWUktVWP/yrzOOZDv+TrRbbRU5GZEVappASIRnh3+6KwkJkjWKE
key_version    1

Request 3
Error writing data to transit/encrypt/test: Error making API request.

URL: PUT http://127.0.0.1:8200/v1/transit/encrypt/test
Code: 429. Errors:

* request path "transit/encrypt/test": rate limit quota exceeded
```

На этот раз не сработал только последний запрос. Если задан более подробный путь, правило квоты ограничения скорости имеет приоритет перед путем.

Если у вас есть другой ключ или вы пытаетесь расшифровать (`transit/decrypt/test`), будет применена квота ограничения скорости `rate-test`.

7. Очистка

Если вы хотите очистить среду после завершения обучения, выполните действия, описанные в этом разделе.

1. Сбросьте переменные окружения `STARVAULT_TOKEN` и `STARVAULT_ADDR`.

```bash
unset STARVAULT_TOKEN STARVAULT_ADDR
```

2. Удалите файлы, созданные во время теста.

```bash
rm test-encryption.sh payload.json audit-payload.json
```

3. Если вы запускаете StarVault локально в режиме `-dev`, вы можете остановить dev-сервер StarVault, нажав Ctrl+C там, где запущен сервер. Или выполните следующую команду.

```bash
pgrep -f starvault | xargs kill
```

2025 orionsoft. Все права защищены.

Настройка производительности

StarVault — это высокопроизводительное решение для управления секретами и защиты данных, способное работать с рабочими нагрузками масштаба корпорации. По мере расширения масштабов использования и внедрения более широких сценариев применения вы можете настраивать StarVault, его базовую операционную систему и хранилище для достижения оптимальной производительности.

Эти рекомендации и лучшие практики помогут вам настроить среду StarVault для достижения оптимальной производительности, но они не предназначены для документирования требований. Это рекомендации, которые следует применять в зависимости от конкретной среды и требований. Руководство также включает важные ограничения ресурсов StarVault, которые следует учитывать в отношении производительности.

Это руководство посвящено настройке среды StarVault для достижения оптимальной производительности. Обратитесь к разделу "Лимиты и максимумы StarVault", чтобы узнать об известных верхних ограничениях на размер определенных полей и объектов, а также о настраиваемых ограничениях для других объектов.

Вы можете сосредоточиться на ограниченном диапазоне настраиваемых параметров, сгруппированных следующим образом:

- Настройка операционной системы охватывает критические элементы конфигурации ОС для идеальной работы.
- Настройка StarVault подробно описывает настройку конфигурации самого StarVault.
- Настройка хранилища содержит элементы, специфичные для хранилища.

Если ваша цель — использовать полученные здесь знания для настройки производственных систем, то вам следует сначала ознакомиться с руководством по эталонной архитектуре и руководством по развертыванию. Убедитесь, что развертывание вашего кластера StarVault соответствует рекомендациям этих ресурсов, прежде чем приступать к выполнению данного руководства. Укрепление боевой среды также является полезным ресурсом для получения информации о защите кластеров для боевой среды.

1. Исследование производительности

Часть настройки производительности включает в себя исследование путем наблюдения и измерения текущих характеристик системы. Для исследования производительности можно

использовать ряд методик и инструментов. Одним из таких методов анализа производительности системы является метод Utilization Saturation and Errors (USE).

В данном методе предлагается методика исследования производительности, которая включает в себя проверку следующих характеристик для каждого соответствующего исследуемого ресурса:

- Утилизация (Utilization) — получали ли вы предупреждения о низком объеме памяти или, например, об ошибках, связанных с нехваткой памяти?
- Насыщенность (Saturation) — есть ли признаки того, что IOPS хранилища достигли допустимого максимума?
- Ошибки (Errors) — есть ли ошибки, например, в логах приложений или логах StarVault? Сохраняются ли они при снижении производительности?

Вы можете применить метод USE к системным ресурсам кластера StarVault и получить более глубокое понимание существующих узких мест или проблем в рамках исследования производительности.

В данном руководстве используются элементы метода USE. Например, при исследовании отказоустойчивости производительности в кластере высокой доступности, ошибки ("E" в USE) могут подсказать, какие ресурсы нуждаются в настройке.

Кроме того, вы можете использовать такие функции, как телеметрия, для сбора показателей и измерения использования и насыщенности ресурсов в кластере StarVault.

Обзор Monitor telemetry & audit device log data позволяет узнать больше об использовании телеметрии StarVault и метрик устройств аудита с помощью стека агрегации на основе Fluentd, Telegraf и Splunk.

Когда вы собираете, исследуйте и измеряйте данные из кластерных сред StarVault, вы также можете более точно обосновать свои решения по настройке производительности.

2. Инструменты для исследования производительности

Метод USE представляет собой всеобъемлющий контрольный список для систем Linux, который отлично подходит для исследования производительности на уровне системы. В методе USE также подробно описаны инструменты, которые можно использовать для исследования использования и насыщения каждого ресурса.

Наиболее распространенные инструменты, которые вы можете использовать для исследования производительности на уровне физической системы или виртуальной машины, также перечислены здесь для справки.

Компонент	Инструмент	Примечание
CPU	dstat, htop, lscpu, sar, top, vmstat	У dstat нет реализации на Python 3; пользователи Red Hat могут эмулировать dstat с помощью Performance Co-Pilot.
Memory	free, sar, vmstat	
Storage	df, iostat, sar, swapon	
Network	ifconfig, netstat	

Для пользователей контейнерных сред, таких как Docker и Kubernetes, существует ряд инструментов более высокого уровня для решения специфических задач по устранению неполадок в этих средах.

К числу широко используемых решений относятся:

- Мониторинг логов данных контейнеров в реальном времени.
- Dynatrace.
- Sysdig Inspect — это мощный интерфейс с открытым исходным кодом для поиска и устранения неисправностей контейнеров.

3. Настройка операционной системы Linux

Ваши развертывания могут получить преимущества от бесперебойной работы StarVault благодаря правильной настройке и конфигурированию базовой операционной системы. В этом разделе вы узнаете о настраиваемой конфигурации ОС Linux для идеальной работы StarVault.

3.1. Пользовательские ограничения

Ядро Linux может устанавливать пользовательские ограничения (также известные как значения `ulimit`) для каждого пользователя, для каждого процесса или для всей системы. Эти ограничения исторически были разработаны для того, чтобы предотвратить потребление всех доступных ресурсов одним пользователем или процессом в многопользовательских и многопроцессных системах.

В современной системе Linux эти ограничения обычно контролируются свойствами процесса systemd.

Для серверов StarVault, на которых установлено минимальное количество запущенных процессов и отсутствуют многопользовательские интерактивные сессии, ограничения по умолчанию могут быть слишком низкими и вызывать проблемы.

Вы можете прочитать активные лимиты для запущенного процесса хранилища из таблицы процессов ядра под соответствующим идентификатором процесса (PID). В этом примере показано использование команды `pidof` для динамического получения PID хранилища и вставки его в путь для получения правильных значений.

```bash
cat /proc/$(pidof starvault)/limits
```

Пример вывода:

```terminal
Limit                     Soft Limit           Hard Limit           Units
Max cpu time              unlimited            unlimited            seconds
Max file size             unlimited            unlimited            bytes
Max data size             unlimited            unlimited            bytes
Max stack size            8388608              unlimited            bytes
Max core file size        0                    unlimited            bytes
Max resident set          unlimited            unlimited            bytes
Max processes             7724                 7724                 processes
Max open files            1024                 4096                 files
Max locked memory         16777216             16777216             bytes
Max address space         unlimited            unlimited            bytes
Max file locks            unlimited            unlimited            locks
Max pending signals       7724                 7724                 signals
Max msgqueue size         819200               819200               bytes
Max nice priority         0                    0
Max realtime priority     0                    0
Max realtime timeout      unlimited            unlimited            us
```

В выводе отображается имя предела и три значения:

- **Мягкий предел (Soft Limit)** — это настраиваемое пользователем значение, которое ядро будет применять, чтобы не превысить жесткий предел.
- **Жесткий предел (Hard Limit)** — это значение, настраиваемое пользователем root, которое будет применяться ядром и которое не может превышать общесистемный предел
- **Единицы (Units)** представляют тип измерения для предела

Хотя в выводе отображаются 16 различных ограничений, в данном руководстве подробно рассматриваются два из них: **Максимальное количество открытых файлов** и **Максимальное количество процессов**.

Будьте осторожны при использовании таких подходов, как `ulimit -a`, для получения значений пользовательских лимитов. Лимиты, выводимые этой командой, предназначены для текущего пользователя и не обязательно совпадают с лимитами идентификатора пользователя, под которым выполнялся ваш процесс StarVault.

3.1.1. Максимальное количество открытых файлов

Операционная система StarVault использует дескрипторы файлов как для доступа к файлам в файловой системе, так и для представления сокетных соединений, установленных с другими сетевыми узлами.

Значение максимального количества открытых файлов, разрешенных процессу StarVault, является критическим пользовательским ограничением, которое следует соответствующим образом настроить для достижения идеальной производительности.

Как измерить использование?

Чтобы узнать текущие значения максимально открытых файлов для процесса хранилища, прочитайте их из таблицы процессов ядра.

BASH
```
cat /proc/$(pidof starvault)/limits | awk 'NR==1; /Max open files/'
```

Пример вывода:

TERMINAL
```
Limit                     Soft Limit           Hard Limit           Units
Max open files            1024                 4096                 files
```

Вы также можете использовать lsof для получения подробной информации об открытых файлах, например как здесь:

BASH
```
sudo lsof -p $(pidof starvault)
```

Пример вывода:

TERMINAL
```
COMMAND       PID      USER   FD      TYPE             DEVICE  SIZE/OFF   NODE
NAME
starvault   14810 starvault   cwd      DIR              253,0      4096      2 /
starvault   14810 starvault   rtd      DIR              253,0      4096      2 /
starvault   14810 starvault   txt      REG              253,0 138377265 131086
/usr/bin/starvault
starvault   14810 starvault    0r      CHR                1,3       0t0      6
/dev/null
starvault   14810 starvault    1u     unix 0xffff89e6347f9c00       0t0  41148
type=STREAM
starvault   14810 starvault    2u     unix 0xffff89e6347f9c00       0t0  41148
type=STREAM
```

```
starvault   14810 starvault    3u     unix 0xffff89e6347f8800       0t0  41208
type=DGRAM
starvault   14810 starvault    4u     a_inode            0,13         0   9583
[eventpoll]
starvault   14810 starvault    6u     IPv4               40467       0t0    TCP
*:8200 (LISTEN)
starvault   14810 starvault    7u     IPv4               41227       0t0    TCP
localhost:53766->localhost:8500 (ESTABLISHED)
```

Это минимальный пример, взятый из только что запечатанного StarVault. Вы можете ожидать гораздо большего результата в StarVault развернутого на боевой среде с несколькими вариантами использования. Результаты полезны для выявления конкретных источников открытых соединений, например, сокетных соединений с механизмом секретов базы данных.

Здесь можно заметить, что последние 2 строки относятся к двум открытым сокетам.

Первый — файловый дескриптор номер 6, открытый с правами чтения и записи (u), имеет тип IPv4, является TCP-узлом, привязанным к порту 8200 на всех сетевых интерфейсах.

Второй — дескриптор файла 7 представляет собой тот же тип сокета, только в виде исходящего эфемерного соединения с портом. Исходящее соединение исходит от StarVault по TCP/53766 к клиентскому агенту Consul на localhost, который прослушивает порт 8500.

Какие типичные ошибки?

Если значение максимального количества открытых файлов слишком мало, StarVault выдает ошибки в оперативный журнал в формате, приведенном в этом примере:

TERMINAL

```
http: Accept error: accept tcp4 0.0.0.0:8200: accept4: too many open files;
retrying in 1s
```

Ключевые части логов:

- Источником ошибки является HTTP-подсистема StarVault (http:).
- Поскольку ошибка исходит от http, она также связана с исчерпанием файловых дескрипторов в контексте сетевых сокетов, а не обычных файлов (обратите внимание на accept4() вместо open()).
- Самая важная часть ошибки и та, которая объясняет непосредственную причину этой проблемы, — **слишком много открытых файлов**. Если максимальное количество открытых файлов не настроено на этом сервере StarVault, то настройка этого значения будет разумной отправной точкой для устранения ошибки.

Эта ошибка является одновременно предупреждением о нехватке дескрипторов файлов и о том, что что-то внутри или вне StarVault может чрезмерно их потреблять.

Проблему следует устранить, увеличив максимальный лимит открытых файлов и перезапустив службу StarVault для каждого затронутого кластерного узла. Увеличение этого значения связано с некоторыми последствиями и ограничениями, о которых вы должны знать, прежде чем делать это.

Существует общесистемный лимит максимально открытых файлов, который устанавливается ядром и который не могут превысить пользовательские программы, такие как StarVault. Обратите внимание, что это значение динамически устанавливается во время загрузки и зависит от физических характеристик системы, таких как доступная физическая память.

Чтобы узнать текущее общесистемное значение максимального количества открытых файлов для данной системы, прочитайте его из таблицы процессов ядра.

BASH

```
cat /proc/sys/fs/file-max
```

Пример вывода:

TERMINAL

```
197073
```

В этой системе невозможно указать максимальный лимит открытых файлов, превышающий 197073.

Увеличение лимитов

В предыдущем примере вы видели, что максимальное количество открытых файлов для процесса StarVault имело мягкое ограничение 1024 и жесткое ограничение 4096. Эти значения часто используются по умолчанию в некоторых дистрибутивах Linux, и для использования StarVault в боевой среде вам всегда следует увеличивать эти значения сверх установленных по умолчанию.

Узнав общесистемный лимит, вы можете соответствующим образом увеличить его для процессов StarVault. В современной Linux на базе systemd это можно сделать, отредактировав файл блока служб StarVault systemd и указав значение для свойства процесса LimitNOFILE.

Имя файла блока systemd может варьироваться, но чаще всего он называется `starvault.service`, и найти его можно по адресу `/lib/systemd/system/starvault.service` или `/etc/systemd/system/starvault.service`.

Редактируйте файл от имени суперпользователя системы:

BASH

```
sudo vi /lib/systemd/system/starvault.service
```

Либо добавьте свойство процесса `LimitNOFILE` в [Service], либо измените его значение, если оно уже существует, чтобы увеличить мягкие и жесткие ограничения до разумного базового значения 65536.

```
LimitNOFILE=65536
```

BASH

Сохраните файл, выйдите из редактора.

Любое изменение устройства требует перезагрузки демона; сделайте это прямо сейчас.

```
sudo systemctl daemon-reload
```

BASH

Эта команда не выводит результатов, если перезагрузка происходит без проблем.

При следующем перезапуске службы хранилища вступят в силу новые ограничения на максимальное количество открытых файлов.

Вы можете перезапустить службу, а затем снова просмотреть таблицу процессов, чтобы убедиться, что изменения вступили в силу.

Следует быть осторожным с этим шагом в системах в боевой среде, поскольку он может изменить лидерство кластера. В зависимости от типа запечатывания StarVault перезапуск службы может означать, что вам также нужно распечатать StarVault, если вы не используете автоматическое запечатывание. Приготовьтесь распечатать StarVault после перезагрузки конфигурации, если не используете автоматическое запечатывание.

Перезапустите службу хранилища.

```
sudo systemctl restart starvault
```

BASH

После перезапуска StarVault проверьте таблицу процессов на наличие нового процесса хранилища:

```
cat /proc/$(pidof starvault)/limits | awk 'NR==1; /Max open files/'
```

BASH

Пример вывода:

```
Limit                     Soft Limit           Hard Limit           Units
Max open files            65536                65536                files
```

TERMINAL

Пример юнит файла systemd StarVault, который также включает это свойство процесса, см. в разделе "Включение и запуск службы" в руководстве по развертыванию StarVault.

3.2. Примечание о масштабировании процессора

Вы можете ожидать, что StarVault будет линейно масштабироваться до 100% использования CPU при настройке определенных рабочих нагрузок, таких как шифрование на движке Transit или Transform Secrets. Как правило, это нереалистичные ожидания.

StarVault создан на языке программирования Go, и отчасти это связано с его характеристиками производительности. В Go есть понятие goroutines — это функции или методы, которые выполняются одновременно с другими функциями или методами.

Чем больше одновременно запланированных goroutines, тем больше переключений контекста выполняет система, тем больше прерываний от сетевого интерфейса и т.д.

Такое поведение может не сильно нагружать процессор с точки зрения его реальной загрузки, но оно может ухудшить работу ввода-вывода. Каждый раз, когда goroutine блокируется для ввода/вывода (или вытесняется из-за прерывания), может пройти больше времени, прежде чем goroutine вернется в работу.

Об этом следует помнить при настройке тяжелых для процессора рабочих нагрузок в StarVault.

4. Настройка StarVault

Следующие разделы относятся к настройке самого программного обеспечения StarVault с помощью доступных параметров конфигурации, возможностей или функциональности.

В этих разделах по мере возможности приводятся рекомендации и примеры.

4.1. Размер кэша

StarVault использует кэш чтения с наименьшим количеством последних использованных записей (LRU) для физической подсистемы хранения с настраиваемым значением cache_-size. Это значение представляет собой количество записей, и по умолчанию оно равно 131072.

Общий размер кэша зависит от размера хранимых записей.

Операции LIST не кэшируются.

4.2. Максимальная продолжительность запроса

StarVault предоставляет два параметра, которые можно настроить, чтобы ограничить максимально допустимую продолжительность запроса. Это можно использовать для

развертывания систем со строгими соглашениями об уровне обслуживания, касающимися продолжительности запросов, или для принудительного установления определенной продолжительности запросов.

На уровне всего сервера есть default_max_request_duration со значением по умолчанию 90 секунд (90s). Опять же, настройка этого значения предназначена для конкретных случаев использования и влияет на каждый запрос, сделанный ко всему узлу, так что имейте это в виду.

Пример минимальной конфигурации StarVault, который показывает использование явного параметра `default_max_request_duration`.

```
api_addr = "https://127.0.0.8200"

default_max_request_duration = "30s"

listener "tcp" {
  address       = "127.0.0.1:8200"
  tls_cert_file = "/etc/pki/starvault-server.crt"
  tls_key_file  = "/etc/pki/starvault-server.key"
}

storage "consul" {
  address = "127.0.0.1:8500"
  path    = "vault"
}
```

Второй вариант — установить аналогичный максимум на уровне слушателей. Вы можете настроить StarVault на использование более чем одного слушателя, добавив несколько блоков слушателей. Чтобы получить некоторую детализацию ограничения запросов, вы можете установить max_request_duration в пределах слушателя. Значение по умолчанию также составляет 90 секунд (90s).

Пример минимальной конфигурации StarVault, который показывает использование явного параметра max_request_duration в TCP-приемнике.

```
api_addr = "https://127.0.0.8200"

listener "tcp" {
  address              = "127.0.0.1:8200"
  tls_cert_file        = "/etc/pki/starvault-server.crt"
  tls_key_file         = "/etc/pki/starvault-server.key"
  max_request_duration = "15s"
}

storage "consul" {
  address = "127.0.0.1:8500"
```

```
  path    = "vault"
}
```

Когда вы задаете `max_request_duration` в блоке TCP listener, это значение отменяет значение `default_max_request_duration`.

4.3. Максимальный размер запроса

StarVault позволяет контролировать глобальный максимально допустимый размер запроса, жестко заданный, в байтах у слушателя с помощью параметра `max_request_size`.

Значение по умолчанию — 33554432 байт (32 МБ).

Указание числа, меньшего или равного 0, полностью отключает ограничение размера запроса.

4.4. Таймауты HTTP

Каждый TCP-слушатель StarVault может определить четыре тайм-аута HTTP, которые напрямую связаны с базовыми параметрами http-сервера Go, определенными в пакете http.

4.4.1. `http_idle_timeout`

Используйте параметр `http_idle_timeout` для настройки максимального времени ожидания следующего запроса при использовании keep-alives. Если значение этого параметра равно 0, StarVault использует значение `http_read_timeout`. Если оба параметра имеют значение 0, таймаут не устанавливается.

Значение по умолчанию: 5m (5 минут)

4.4.2. `http_read_header_timeout`

Вы можете использовать параметр `http_read_header_timeout`, чтобы настроить количество времени, отведенное на чтение заголовков запроса. Если значение http_read_-header_timeout равно 0, StarVault использует значение `http_read_timeout`. Если оба значения равны 0, таймаут отсутствует.

Значение по умолчанию: 10 с (10 секунд)

4.4.3. `http_read_timeout`

С помощью параметра `http_read_timeout` можно настроить максимальную продолжительность чтения всего HTTP-запроса, включая тело.

Значение по умолчанию: 30 с (30 секунд).

4.4.4. `http_write_timeout`

С помощью параметра `http_write_timeout` можно настроить максимальную продолжительность тайминга записи ответа.

Значение по умолчанию: 0 (ноль)

4.5. Истечение срока аренды и значения TTL

StarVault поддерживает аренды для всех динамических секретов и токенов аутентификации сервисного типа.

Эти аренды представляют собой обязательства по выполнению будущей работы в форме отзыва, которая включает в себя подключение к внешним узлам для отзыва учетных данных там же. Кроме того, StarVault должен выполнять внутреннее обслуживание в виде удаления (потенциально рекурсивно) просроченных токенов и аренд.

Важно следить за ростом числа аренд в кластере StarVault в боевой среде. Неограниченный рост аренды может привести к серьезным проблемам с базовым хранилищем и, в конечном счете, с самим StarVault.

По умолчанию StarVault использует значение time-to-live (TTL) 32 дня для всех аренд. Вы должны помнить об этом при определении сценариев использования и стараться выбирать как можно более короткое значение TTL, которое вы можете использовать.

Если вы развертываете StarVault без указания явных значений TTL и максимального TTL, вы рискуете сгенерировать чрезмерное количество аренд, поскольку TTL по умолчанию позволяет им легко накапливаться. Массовая или нагрузочная генерация и тестирование усиливают этот эффект. Это часто встречается у новых пользователей StarVault. Ознакомьтесь с разделами "Время жизни токена", "Периодические токены" и "Явные максимальные TTL", чтобы узнать больше.

4.5.1. Короткие TTL

Короткие TTL — хорошо для безопасности.

- Скомпроментированный токен с коротким временем аренды истекает раньше.
- Сбой или уничтожение экземпляра службы, чей токен не был отозван вскоре после использования, не является большой проблемой, если он все равно имеет короткий TTL.

Короткие TTL — хорошая производительность.

Короткие TTL имеют эффект сглаживания нагрузки. Лучше иметь много небольших записей, разнесенных во времени, чем сразу большое количество просроченных аренд.

4.5.2. На что обратить внимание

Что касается использования и перенасыщение, вы можете выявить проблемы, отслеживая метрику `starvault.expire.num_leases`, которая представляет собой количество всех аренд, срок действия которых подходит к концу.

Можно также отслеживать емкость хранилища на предмет признаков перенасыщения арендами. В частности, можно изучить пути в хранилище, в которых хранятся данные об аренде. Просмотрите руководство "Проверка данных в хранилище Consul" или "Проверка данных в интегрированном хранилище", чтобы узнать больше о путях, где можете найти данные об аренде.

4.6. Сертификаты PKI и списки отзыва сертификатов

Пользователи PKI Secrets Engine должны знать о соображениях производительности и лучших практиках, характерных для этого механизма секретов.

Для максимальной производительности этого механизма секретов, следует учитывать следующее: пределы производительности зависят от доступной энтропии на сервере StarVault и высоких требований к процессору для вычисления пар ключей. Если в вашем случае StarVault выпускает сертификаты и ключи, а не подписывает запросы на подписание сертификатов (CSR).

Это может привести к линейному масштабированию. Наиболее универсальный способ избежать этого — заставить клиентов генерировать CSR и отправлять их в StarVault для подписания, а не заставлять StarVault возвращать пару сертификат/ключ.

Две наиболее распространенные проблемы, с которыми сталкиваются пользователи при работе с механизмом секретов PKI, связаны друг с другом и могут привести к серьезным сбоям в работе. В крайних случаях эти проблемы могут привести к полному отключению хранилища.

Первая проблема связана с выбором нереалистично долгого времени жизни сертификата.

StarVault придерживается философии, согласно которой время жизни всех секретов должно быть настолько коротким, насколько это практически возможно. Хотя это замечательно с точки зрения безопасности, выбор идеальных значений срока действия сертификатов может быть несколько затруднен.

По-прежнему важно тщательно продумать каждый случай использования и определить идеальное минимальное время жизни для секретов StarVault, включая сертификаты PKI, созданные StarVault. Чтобы узнать больше, просмотрите документацию по механизму PKI-секретов, сосредоточившись на разделе "Поддерживайте короткие сроки жизни сертификатов, ради CRL".

Если время жизни сертификата несколько превышает требуемое, важно убедиться, что приложения используют сертификаты, полученные из StarVault, до истечения срока их действия, прежде чем запрашивать новые, и не запрашивают их на регулярной основе. Долгоживущие сертификаты часто становятся причиной быстрого роста CRL.

Вторая проблема является симптомом первой, поскольку создание нескольких сертификатов с длительным сроком службы приводит к быстрому росту списка отзыва сертификатов (Certificate Revocation List, CRL). Этот список внутренне представлен как один ключ в хранилище ключей/значений. Если ваши серверы StarVault используют хранилище Consul, оно поставляется с максимальным размером значения по умолчанию в 512 КБ. Со временем CRL может насытиться этим значением при достаточно неправильном использовании и частом запросе сертификатов с большим сроком действия.

Типичные ошибки

Если CRL механизма секретов PKI стал больше, чем допускается максимальным размером значения ключа Consul по умолчанию, вы можете столкнуться с ошибками об отзыве аренды в операционном журнале StarVault, похожими на этот пример:

TERMINAL

```
[ERROR] expiration: failed to revoke lease:
lease_id=pki/issue/prod/7XXYS4FkmFq8PO05En6rvm6m error="failed to revoke entry:
resp: (*logical.Response)(nil) err: error encountered during CRL building: error
storing CRL: Failed request: Request body too large, max size: 524288 bytes"
```

Если вы хотите увеличить производительность механизма секретов PKI и не нуждаетесь в CRL, вам следует определить свои роли так, чтобы они использовали параметр no_store.

StarVault не может перечислить или отозвать сертификаты, созданные ролями, для которых определен параметр `no_store`.

4.7. ACL в политиках

Если вашей целью является максимальная оптимизация производительности StarVault, вам следует проанализировать политики ACL и пути политик, чтобы минимизировать сложность путей, использующих шаблоны и специальные операторы.

4.7.1. Повышение производительности

- Старайтесь по возможности минимизировать использование шаблонов в путях политики.
- Старайтесь свести к минимуму использование обозначений сегментов пути `+` и `*` в синтаксисе пути политики.

4.8. Устройства аудита

Убедитесь, что ваши устройства аудита могут беспрепятственно выполнять запись, а также не забудьте настроить целевое назначение устройства. Например, следует настроить хранилище, используемое устройством аудита файлов, так, чтобы оно могло работать с максимальным потенциалом.

Ознакомьтесь с документацией по исключению аудита, чтобы узнать больше о том, как работает исключение из устройства аудита.

4.9. Оценка политики

Для справки здесь приведены диаграмма и описание процесса оценки политик StarVault для ACL, EGP и RGP.

Если запрос был неаутентифицированным (например, `starvault login`), токен отсутствует, поэтому StarVault оценивает EGP, связанные с конечной точкой запроса.

Если в запросе есть токен, оцениваются политики ACL, прикрепленные к токену.

Если маркер имеет соответствующие возможности для работы на пути, StarVault оценивает RGPs. Затем StarVault оценивает EGPs, установленные на конечной точке запроса.

Если на каком-либо этапе оценка политики не удается, StarVault отклоняет запрос.

4.10. Токены

StarVault требует действительных токенов для всех аутентифицированных запросов, к которым относится большинство конечных точек API.

Обычно они имеют ограниченное время жизни в виде срока аренды или времени жизни (TTL).

Чаще всего токены используются для запросов на вход в систему и для отзыва. Эти взаимодействия с StarVault приводят к следующим операциям.

Взаимодействие	**Операции StarVault**
Запрос на вход в систему	Запись нового токена в хранилище токенов Запись нового договора аренды в хранилище договоров аренды
Отзыв токена (или истечение срока действия токена)	Удалить токен Удалить аренду токена Удалить все дочерние токены и аренды

Пакетные токены — это зашифрованные двоичные большие объекты, которые содержат достаточно информации, чтобы StarVault мог использовать их для выполнения действий, но не требуют хранения на диске, как сервисные токены.

При использовании пакетных токенов следует помнить о некоторых компромиссах и применять их с осторожностью.

4.10.1. Менее безопасно, чем сервисные токены

- StarVault не может отзывать или обновлять пакетные токены.
- Вы должны заранее задать значение TTL, и в результате оно часто оказывается выше идеального.

4.10.2. Лучшая производительность

- Пакетные токены недороги в использовании, поскольку они не работают с диском.
- Они часто являются приемлемым компромиссом, когда альтернативой является неуправляемая частота запросов на вход.

5. Настройка системы хранения

Задержка запросов StarVault в основном ограничивается настроенным типом хранилища, а запись в хранилище обходится гораздо дороже, чем чтение.

Большинство операций записи в StarVault связано с этими событиями:

- Вход в систему и создание токенов.
- Динамическое создание секретов.
- Продление.
- Отзыв.

Существует ряд аналогичных настраиваемых параметров для поддерживаемых хранилищ. В этом руководстве рассматриваются параметры для Integrated Storage (Raft) и Consul.

Существуют некоторые эксплуатационные характеристики и компромиссы, связанные с тем, как различные механизмы хранения обрабатывают память, персистентность и сетевые возможности, с которыми вам следует ознакомиться.

Характеристики хранилища Consul:

Хранилище	**Примечание**
Consul	Consul имеет лучшую производительность записи на диск, чем Integrated Storage.

Хранилище	Примечание
Плюсы	Рабочий набор находится в памяти, поэтому он отличается высокой производительностью.
Минусы	Эксплуатационные сложности Сложнее отлаживать и устранять неисправности Задействованы сетевые переходы, теоретически выше сетевая задержка Более частые снапшоты влияют на производительность Ограничение памяти с более высокой вероятностью возникновения ситуаций, связанных с выходом за пределы памяти

Характеристики хранилища Integrated Storage (Raft):

Хранилище	Примечание
Raft	Integrated Storage (Raft) имеет лучшую сетевую производительность, чем хранилище Consul.
Плюсы	Операционно проще Менее частые снапшоты, так как данные сохраняются на диске Нет сетевых переходов (как компромисс — дополнительная запись `fsync()` в BoltDB в менеджере конечных состояний)
Минусы	Данные сохраняются на диске, поэтому теоретически производительность при записи несколько ниже Производительность записи немного ниже, чем у Consul

Учитывая эту информацию, просмотрите подробную информацию о конкретных настраиваемых параметрах для хранилища, которое вас больше всего интересует.

5.1. Consul

При использовании Consul в качестве хранилища большая часть дисковых операций ввода-вывода приходится на серверы Consul, а сам StarVault по сравнению с ними использует гораздо меньше дисковых операций ввода-вывода. Consul хранит свой рабочий набор в памяти. По общему правилу, сервер Consul должен иметь физическую память, равную примерно 3-кратному размеру рабочего набора данных хранилища ключей/значений, содержащего данные StarVault. Крайне важно поддерживать хорошую производительность операций ввода-вывода в секунду (IOPS) для хранилища Consul. Для получения более

подробной информации ознакомьтесь с эталонной архитектурой Consul и руководством по развертыванию Consul.

5.1.1. Типичные ошибки

Если вы наблюдаете сильное снижение производительности StarVault при использовании Consul для хранения данных, в первую очередь обратите внимание на использование памяти и ошибки сервера Consul. Например, проверьте кольцевой буфер ядра операционной системы сервера Consul или syslog на наличие признаков нехватки памяти (OOM).

```bash
grep 'Out of memory' /var/log/messages
```

Если результаты есть, они будут похожи на этот пример.

```terminal
kernel: [16909.873984] Out of memory: Kill process 10742 (consul) score 422 or
sacrifice child
kernel: [16909.874486] Killed process 10742 (consul) total-vm:242812kB, anon-
rss:142081kB, file-rss:68768kB
```

Снижение IOPS на серверах Consul — еще одна распространенная причина проблем. Это состояние может проявляться в StarVault в виде ошибок, связанных с отменой контекста, как в следующих примерах.

```terminal
[ERROR] core: failed to create token: error="failed to persist entry: context
canceled"

[ERROR] core: failed to register token lease: request_path=auth/approle/login
error="failed to persist lease entry: context canceled"

[ERROR] core: failed to create token: error="failed to persist accessor index
entry: context canceled"
```

Ключевым моментом здесь является сообщение "context canceled". Эта проблема приведет к прерывистой доступности StarVault для всех пользователей, и вы должны попытаться устранить ее, увеличив доступное количество IOPS для серверов Consul.

5.1.2. Параметры Consul связанные с производительностью

Ниже перечислены важные параметры конфигурации, связанные с производительностью, о которых должны знать при использовании Consul для хранилища StarVault.

5.1.3. `kv_max_value_size`

Одним из распространенных ограничений производительности, с которым можете столкнуться при использовании Consul как хранилище для StarVault, является размер

данных, которые StarVault может записать в качестве значения одного ключа в хранилище ключей/значений Consul.

Начиная с версии Consul 1.7.2 можно явно указать это значение в байтах с помощью конфигурационного параметра `kv_max_value_size`.

Значение по умолчанию: 512 КБ

Вот пример фрагмента конфигурации сервера Consul, который увеличивает это значение до 1024 КБ.

```
"limits": {
    "kv_max_value_size": 1024000
}
```

StarVault возвращает следующую ошибку клиенту, который пытается превысить максимальный размер значения.

TERMINAL

```
Error writing data to kv/data/foo: Error making API request.

URL: PUT http://127.0.0.1:8200/v1/kv/data/foo
Code: 413. Errors:

* failed to parse JSON input: http: request body too large
```

Неправильная настройка этого параметра может привести к неожиданным сбоям в работе Consul, потенциально повлиять на стабильность и предотвратить регулярные сигналы от значительного увеличения длительности ввода-вывода RPC.

5.1.4. `txn_max_req_len`

Этот параметр задает максимальное количество байт для тела запроса транзакции к конечной точке Consul `/v1/txn`. В ситуациях, когда вы задаете и `txn_max_req_len`, и `kv_max_value_size`, более высокое значение имеет приоритет для обоих параметров.

Неправильная настройка этого параметра может привести к неожиданным сбоям в работе Consul, потенциально повлиять на стабильность и предотвратить регулярные сигналы от значительного увеличения длительности ввода-вывода RPC.

5.1.5. `max_parallel`

Параметр max_parallel — еще один параметр, который иногда полезно настраивать в зависимости от конкретного окружения и конфигурации. Параметр max_parallel задает максимальное количество параллельных запросов, которые StarVault может делать к Consul.

По умолчанию это значение равно 128.

Это значение обычно не увеличивают для повышения производительности, чаще всего его используют для снижения нагрузки на перегруженный кластер Consul путем уменьшения значения по умолчанию.

5.1.6. `consistency_mode`

StarVault поддерживает использование 2 из 3 режимов согласованности Consul. По умолчанию он использует режим по умолчанию, который в документации Consul описывается следующим образом:

```
Если не указано, то по умолчанию в большинстве случаев используется строгая
согласованность. Однако существует небольшое окно, когда StarVault может избрать
нового лидера, в течение которого старый лидер может обслуживать устаревшие
значения. Компромисс заключается в быстром чтении, но потенциально неактуальных
значениях. Условие, приводящее к pass:[<span data-
tooltip="stale">неактуальному</span>] чтению, трудно вызвать, и большинство
клиентов не должны беспокоиться об этом случае. Также обратите внимание, что это
состояние гонки применимо к чтению, но не к записи.
```

Этот режим подходит для большинства случаев использования, и вы должны знать, что изменение режима на **strong** в StarVault отображает последовательный режим в Consul. Этот режим имеет больше последствий для производительности, и в большинстве случаев он не нужен, если только вы не переносите чтение хранилища. В документации Consul говорится следующее о последовательном режиме:

```
Этот режим является строго последовательным без каких-либо оговорок. Он требует,
чтобы лидер подтвердил кворуму, что он по-прежнему является лидером. Это влечет
за собой дополнительный обход всех серверов. Увеличение задержки является
компромиссом из-за дополнительной поездки туда и обратно. Большинство клиентов
не должны использовать этот метод, если только они не могут терпеть pass:[<span
data-tooltip="stale">неактуальное</span>] чтение.
```

5.2. Integrated Storage (Raft)

В StarVault есть возможность Integrated Storage, которая использует Raft Storage. По своему поведению и набору функций Integrated Storage очень похоже на хранилище ключей/значений Consul. Оно реплицирует данные StarVault на все серверы, используя алгоритм консенсуса Raft.

Если вы еще этого не сделали, просмотрите контрольный список миграции для получения дополнительной информации об Integrated Storage.

Ниже перечислены настраиваемые элементы конфигурации для Integrated Storage.

5.2.1. `mlock()`

Отключите функцию `mlock()`, если в развертывании StarVault используется Integrated Storage. Integrated Storage плохо взаимодействует с файлами, отображаемыми в памяти, такими как файлы, созданные BoltDB, которые Raft использует для отслеживания состояния.

При использовании функции `mlock()` файлы, сопоставленные с памятью, загружаются в резидентную память, что приводит к загрузке всего набора данных StarVault в память, а это может привести к выходу за пределы памяти, если объем данных StarVault превысит объем доступной физической памяти.

5.2.1.1. Рекомендации

Хотя данные StarVault в BoltDB остаются зашифрованными в состоянии покоя, рекомендуется использовать инструкции для вашей ОС, чтобы отключить swap на серверах StarVault, использующих Integrated Storage, чтобы предотвратить запись ОС на диск конфиденциальных данных StarVault, хранящихся в памяти.

5.2.1.2. Типичные ошибки

Если вы работаете с кластером StarVault с Integrated Storage и не отключили функцию `mlock()` для бинарного файла starvault (и, возможно, всех внешних плагинов), то вы можете наблюдать ошибки, подобные этой, когда данные StarVault превышают доступный объем памяти.

TERMINAL

```
kernel: [12209.426991] Out of memory: Kill process 23847 (starvault) score 444
or sacrifice child
kernel: [12209.427473] Killed process 23847 (starvault) total-vm:1897491kB,
anon-rss:948745kB, file-rss:474372kB
```

5.2.2. `performance_multiplier`

StarVault использует конфигурационный параметр `performance_multiplier` аналогичным образом в контексте Integrated Storage для масштабирования ключевых временных параметров алгоритма Raft.

Значение по умолчанию равно 0.

Настройка этого параметра влияет на время, которое требуется StarVault для обнаружения сбоев лидера и проведения выборов лидера, за счет того, что для повышения производительности требуется больше сетевых ресурсов и ресурсов процессора.

По умолчанию StarVault использует менее производительную синхронизацию, которая подходит для серверов StarVault со скромными ресурсами. Настройка по умолчанию равна установке значения 5. Установка значения 1 переводит Raft в режим максимальной производительности, рекомендуемый для производственных серверов StarVault. Максимально допустимое значение — 10.

Это значение по умолчанию может измениться в будущих версиях StarVault, если целевой минимальный профиль сервера изменится.

5.2.3. `snapshot_threshold`

Это низкоуровневый параметр, который редко нуждается в настройке.

Опять же, параметр `snapshot_threshold` похож на тот, с которым вы могли сталкиваться при развертывании Consul. Если вы не знакомы с Consul, он автоматически делает снимки данных фиксации raft. Параметр `snapshot_threshold` управляет минимальным количеством записей коммита raft между снимками, сохраняемыми на диск.

В документации говорится следующее о настройке этого параметра:

```
Занятые кластеры, испытывающие избыточный дисковый ввод-вывод, могут увеличить
это значение, чтобы уменьшить дисковый ввод-вывод и минимизировать вероятность
того, что все серверы будут делать снимки одновременно. При увеличении этого
значения происходит обмен дискового ввода-вывода на дисковое пространство,
поскольку журнал будет расти намного больше, а StarVault не сможет освободить
место в raft.db до следующего моментального снимка. При увеличении этого
значения серверы могут дольше восстанавливаться после сбоев или обхода отказа,
поскольку StarVault придется воспроизводить больше журналов.
```

6. Ограничения и максимумы ресурсов

6.1. Максимальное количество механизмов секретов

Не существует конкретного ограничения на количество включенных механизмов секретов.

В зависимости от типа используемого хранилища, при наличии нескольких тысяч (потенциально десятков тысяч) включенных механизмов секретов StarVault может ограничить (например) максимальный размер значения.

6.2. Максимальный размер значения при использовании хранилища Consul

По умолчанию максимальный размер значения для ключа в хранилище ключей/значений Consul соответствует предложенному Raft максимальному размеру в 512 КБ. Начиная с версии Consul 1.7.2, вы можете изменить это ограничение с помощью `kv_max_value_size`.

6.3. Максимальный размер значения при использовании Integrated Storage

В отличие от хранилища Consul, Integrated Storage не устанавливает максимальный размер значения ключа. Это означает, что вам следует проявлять осторожность при развертывании на Integrated Storage сценариев использования, которые могут привести к неограниченному росту значения.

Integrated Storage не так сильно зависит от памяти и подвержено ее нехватке из-за того, как StarVault сохраняет данные на диск. Тем не менее, использование слишком больших значений для ключей может негативно сказаться на координации сети, голосовании и выбора лидеров. Помните, что StarVault Integrated Storage не предназначена для работы в качестве базы данных ключей/значений общего назначения. Если вы используете ключи с неоправданно большими значениями (в несколько раз больше, чем по умолчанию), это может привести к проблемам, в зависимости от вашего случая использования и окружения.

2025 orionsoft. Все права защищены.

История изменений

История изменений отражает хронологию развития StarVault: расширение функциональности, изменения, исправления ошибок и другие обновления.

v1.2.0

Август 2025

Расширение функциональности

1. Добавлен новый метод userpass-advanced, позволяющий:
   - использовать принудительную смену пароля при первом входе;
   - выполнять ручную смену пароля пользователями;
   - настраивать политики генерации паролей (минимальная длина, charsets и пр.).
2. Добавлен метод аутентификации LDAP Meta с возможностью маппинга и синхронизации пользовательских атрибутов в метаданные Entity. В OIDC-провайдер добавлена поддержка пользовательских атрибутов, доступных для использования в маппингах и политике.

v1.1.0

▶ Описание обновления

2025 orionsoft. Все права защищены.