?????????? ???????? ??????????

Дата обновления: 23.04.2026

В данной статье на XDR описаны общие рекомендации, которые также применимы к функциональности KEDR. Нас интересует именно требования к дискам и хранилищам.

Перед дальнейшими изменениями можете выполнить команду на сервере СУБД и записать результаты:

```
fio --name=test --directory=/var/lib/postgresql/<ваша_версия>/main --rw=randrw --bs=8k --size=1G --iodepth=16 --ioengine=libaio --direct=1 --fsync=1 --runtime=60
```

Для начала выполним быструю проверку типа диска:

```
lsblk -d -o name,rota,TYPE,MODEL,SIZE
```

Ожидаемые выводы и их интерпритация:

```
#Пример 1: SSD
NAME    ROTA    TYPE    MODEL             SIZE
sda        0    disk    Samsung_SSD_860    500G   # ROTA=0 (SSD)

#Пример 2: HDD
NAME    ROTA    TYPE    MODEL             SIZE
sda        1    disk    WDC_WD10EZEX       1T     # ROTA=1 (HDD)

#Пример 3: Виртуальный диск (VMware)
NAME    ROTA    TYPE    MODEL             SIZE
sda        0    disk    Virtual_disk       100G   # ROTA=0 (считаем как SSD)

#Пример 4: NVMe
NAME    ROTA    TYPE    MODEL             SIZE
nvme0n1    0    disk    INTEL_SSD          500G   # NVMe (всегда SSD)
```

Далее настроим планировщие I/O:

Создадим udev правило и отредактируем его:

```
nano /etc/udev/rules.d/60-scheduler.rules
```

Для SSD (ROTA=0) вставим следующие значения:

```
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0",
ATTR{queue/scheduler}="mq-deadline"
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0",
ATTR{queue/nr_requests}="1024"
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0",
ATTR{queue/max_sectors_kb}="2048"
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0",
ATTR{queue/read_ahead_kb}="128"
```

Для HDD (ROTA=1):

```
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="1",
ATTR{queue/scheduler}="mq-deadline"
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="1",
ATTR{queue/nr_requests}="256"
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="1",
ATTR{queue/max_sectors_kb}="512"
ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="1",
ATTR{queue/read_ahead_kb}="256"
```

Для NVMe (ROTA=0):

```
ACTION=="add|change", KERNEL=="nvme[0-9]*", ATTR{queue/scheduler}="none"
ACTION=="add|change", KERNEL=="nvme[0-9]*", ATTR{queue/nr_requests}="1024"
```

И применим правила:

```
udevadm control --reload-rules
udevadm trigger --name-match=sda #изменить имя диска, если отличается
```

Далее настроим параметры ядра (sysctl):

Создадим файл настроек и отредактируем его:

```
nano /etc/sysctl.d/99-database-tuning.conf
```

Для SSD / NVMe / Виртуальных дисков (ROTA=0):

```
vm.dirty_ratio = 10
vm.dirty_background_ratio = 5
vm.dirty_expire_centisecs = 3000
vm.dirty_writeback_centisecs = 500

fs.aio-max-nr = 2097152

vm.swappiness = 1
vm.vfs_cache_pressure = 50

net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

Для HDD (ROTA=1):

```
vm.dirty_ratio = 30
vm.dirty_background_ratio = 10
vm.dirty_expire_centisecs = 6000
vm.dirty_writeback_centisecs = 3000

fs.aio-max-nr = 2097152

vm.swappiness = 1
vm.vfs_cache_pressure = 50

net.core.rmem_max = 16777216
net.core.wmem_max = 16777216
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
```

И применем настройки:

```
sysctl -p /etc/sysctl.d/99-database-tuning.conf
```

Далее настроим опции файловой системы:

Отредактируем fstab (предварительно сделайте резервную копию):

```
nano /etc/fstab
```

Найдём строку, как пример для ext4:

```
UUID=3fe51788-bb69-467d-89a9-a146c1df7fdd /                       ext4    defaults        1 1
```

Заменим на:

```
UUID=3fe51788-bb69-467d-89a9-a146c1df7fdd /                       ext4
rw,noatime,nodiratime,data=ordered,errors=remount-ro       1 1
```

Для xfs заменим на: **noatime,nodiratime,logbufs=8,logbsize=256k**

Далее перемонтируем корневую систему:

```
mount -o remount /
```

или перезагрузим систему.

Также рекомендую обновить systemd после изменения /etc/fstab:

```
systemctl daemon-reload
```

Далее настроим TRIM (только для SSD):

```
# Включить fstrim.timer:
systemctl enable --now fstrim.timer

# Проверить статус:
systemctl status fstrim.timer
```

Далее перепроверим конфигуарцию postgresql.conf:

```
nano /etc/postgresql/<ВЕРСИЯ>/main/postgresql.conf

# Переопределите следующие параметры:
listen_addresses = '*'
port = 5432
max_connections = 512
shared_buffers = 8GB                  # 25% от ОЗУ, минимум 3 ГБ
effective_cache_size = 24GB           # 75% от ОЗУ
temp_buffers = 24MB
```

```
work_mem = 64MB
maintenance_work_mem = 1GB
max_stack_depth = 7MB                 # Для Linux: ulimit -s минус 1 МБ
effective_io_concurrency = 200        # 200 для SSD или 2 для HDD
max_parallel_workers_per_gather = 0
wal_buffers = 64MB
max_wal_size = 4GB
min_wal_size = 1GB
random_page_cost = 1.1                # 1.1 для SSD или 4.0 для HDD
log_hostname = 1
standard_conforming_strings = on      # Обязательно должно быть 'on'
```

Повторно можете выполнить команду fio и сравнить с первоначальными результатами. Должны увидеть рост пропускной способности, уменьшение задержек, увеличение числа операций в единицу времени.

Revision #4
Created 27 March 2026 10:15:25 by Кирилл
Updated 14 July 2026 08:25:57 by Кирилл