Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Тюнинг ядра Linux для высоконагруженных серверов

Sysctl-профили под роль сервера: сеть, дисковый I/O, память — от baseline до контроля дрейфа

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Под нагрузкой (веб-фронт, 1С, PostgreSQL, файловые сервисы) сервер заказчика «упирается» задолго до пределов железа: дефолты ядра Linux рассчитаны на универсальную машину, а не на десятки тысяч соединений и NVMe. Итог — отброшенные SYN в пике, рост latency, секундные фризы на записи и OOM-kill в разгар продаж. Мы приводим ядро к роли сервера управляемыми sysctl-профилями с зам…

Почему это важно бизнесу

- Пики нагрузки (акции, отчётный период) роняют сервис при исправном железе — теряются заказы и часы работы всего штата
- Деградация СУБД — это медленная 1С и касса: минуты ожидания на каждом документе умножаются на всех сотрудников
- Апгрейд железа не помогает: узкое место — очереди ядра, деньги уходят в простаивающие CPU и RAM
- Ручные правки «по статье из интернета» без учёта дают нестабильность и невозможность найти причину сбоя

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

65535

net.core.somaxconn и tcp_max_syn_backlog: очередь соединений под пики (дефолт somaxconn — 4096…

по докам docs.kernel.org, ip-sysctl

bbr + fq

tcp_congestion_control=bbr и default_qdisc=fq на интернет-фронтах; в чистой LAN оставляем cubic

docs.kernel.org, модуль tcp_bbr (с 4.9)

16 МБ

Потолок net.core.rmem_max/wmem_max и tcp_rmem/tcp_wmem под 10G-аплинк; на 1G достаточно 4–8 МБ

наш стандарт, расчёт по BDP

5% / 15%

vm.dirty_background_ratio/dirty_ratio — ранний фоновый flush вместо секундных stall записи

по докам docs.kernel.org, sysctl/vm

never

transparent_hugepage=never для СУБД; вместо THP — статические страницы vm.nr_hugepages

наш стандарт для PostgreSQL/Redis

1

vm.swappiness на серверах БД (диапазон 0–200 с ядра 5.8); swap оставляем как страховку от OOM-…

по докам docs.kernel.org, sysctl/vm

РЕАЛИЗАЦИЯ 1 ИЗ 3

Сетевой стек: профиль 90-network.conf на фронтах

Что настраиваем

Nginx/HAProxy-фронты и reverse-proxy клиентов; Ubuntu 22.04/24.04, ядра 5.15–6.8, аплинки 1–10G

Как мы это делаем

1. Снимаем baseline: ss -s, nstat -az (TcpExtListenDrops/ListenOverflows), sar -n DEV; фиксируем p99 через wrk до правок
2. Drop-in /etc/sysctl.d/90-network.conf: net.core.somaxconn=65535, net.ipv4.tcp_max_syn_backlog=65535, net.core.netdev_max_backlog=50000
3. Буферы по BDP канала: net.core.rmem_max/wmem_max=16777216 для 10G; в tcp_rmem/tcp_wmem поднимаем только третье значение (max)
4. BBR: modprobe tcp_bbr, net.ipv4.tcp_congestion_control=bbr + net.core.default_qdisc=fq; в LAN-only сегментах оставляем cubic
5. Поднимаем backlog в приложении (nginx: listen ... backlog=65535), применяем sysctl --system, сверяем wrk с baseline

РЕЗУЛЬТАТ

Фронты держат 50–100 тыс. одновременных соединений без ListenDrops и роста p99; на каналах с потерями BBR даёт кратный выигрыш полосы против CUBIC; каждый шаг подтверждён замером, поэтому эффект атрибутируем и воспроизводим на следующем внедрении.

КЛЮЧЕВОЙ НЮАНС

somaxconn — только потолок: listen() приложения должен запросить такой же backlog. tcp_tw_reuse на современных ядрах по умолчанию 2 (только loopback), а tcp_tw_recycle удалён из ядра 4.12 — не ищем его.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Дисковый I/O и память под СУБД (PostgreSQL, 1C)

Что настраиваем

Серверы PostgreSQL/СУБД под 1C и веб: NVMe и SATA SSD, 64–256 ГБ RAM, включая двухсокетные NUMA-платформы

Как мы это делаем

1. Udev-правило 60-scheduler.rules: планировщик none для NVMe, mq-deadline для SATA SSD, bfq для HDD — переживает ребут и hotplug
2. 91-io.conf: vm.dirty_background_ratio=5, vm.dirty_ratio=15, vm.dirty_expire_centisecs=1500 — размазываем flush вместо секундных stall
3. THP гасим параметром ядра transparent_hugepage=never в GRUB_CMDLINE_LINUX — исключаем latency-спайки от compaction
4. Статические hugepages: vm.nr_hugepages из расчёта shared_buffers / 2 МБ + запас; в postgresql.conf — huge_pages=on с контролем старта
5. На двухсокетных платформах пиннинг через systemd: NUMAPolicy=bind, NUMAMask=0; топологию сверяем numactl --hardware

РЕЗУЛЬТАТ

СУБД перестаёт «замирать» на checkpoint и compaction: latency записи выравнивается, TLB-промахи на больших shared_buffers падают на порядок, pgbench стабильно показывает прирост TPS в десятки процентов на том же железе.

КЛЮЧЕВОЙ НЮАНС

swappiness=1 не выключает swap и не должен — это страховка от OOM. На больших RAM берём vm.dirty_bytes вместо *_ratio: 15% от 256 ГБ — это 38 ГБ грязных страниц, flush такого объёма сам станет stall.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Тираж профилей и контроль дрейфа по парку

Что настраиваем

Весь Linux-парк на обслуживании: Ubuntu/Debian/AlmaLinux, роль sysctl-profiles в Ansible, метрики в Zabbix/Prometheus

Как мы это делаем

1. Профили по ролям (front/db/file) храним в git; Ansible-роль раскладывает /etc/sysctl.d/9*-*.conf и запускает sysctl --system
2. Дескрипторы сквозной цепочкой: fs.nr_open → DefaultLimitNOFILE в system.conf → LimitNOFILE=262144 в юните сервиса
3. Мониторим TcpExtListenOverflows, TCP-ретрансмиты, swap in/out и iowait; триггеры на рост drop-ов и очередей
4. Ежесуточная сверка sysctl -a с эталоном профиля — ручные echo и пакетные твики видны в отчёте как дрейф
5. После обновления ядра — регрессионный прогон fio/iperf3/pgbench и ревизия профиля по изменениям дефолтов

РЕЗУЛЬТАТ

Каждый сервер воспроизводим: профиль восстанавливается одним прогоном роли, ручные правки местных админов «по статье» видны в суточном отчёте как дрейф, а деградации ловятся триггером по drop-ам раньше, чем их заметят пользователи.

КЛЮЧЕВОЙ НЮАНС

Имена и дефолты sysctl меняются между ядрами (somaxconn 128→4096 в 5.4, диапазон swappiness до 200 с 5.8) — профиль без ревизии после апгрейда дистрибутива превращается в карго-культ.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Слепое копирование чужого sysctl**
  Буферы 16 МБ на VPS со 100 Мбит съедают память впустую; считаем BDP канала и тюним только под свой профиль нагрузки
- ✕ **Поиск tcp_tw_recycle**
  Параметр удалён из ядра 4.12 — ломал клиентов за NAT. tcp_tw_reuse касается только исходящих и с дефолтом 2 работает лишь на loopback
- ✕ **THP выключен только echo в /sys**
  После ребута enabled возвращается в madvise/always; фиксируем transparent_hugepage=never в cmdline ядра через GRUB
- ✕ **somaxconn поднят, приложение — нет**
  Реальная очередь = min(somaxconn, аргумент listen()); поднимаем и в nginx (listen backlog=), и в Redis (tcp-backlog)
- ✕ **swappiness=0 как «выключить swap»**
  Ядро всё равно свопит при нехватке памяти, а риск OOM-kill растёт; ставим 1 и держим swap как амортизатор
- ✕ **Планировщик I/O задан через echo**
  Значение в /sys/block/*/queue/scheduler слетает при ребуте и hotplug; закрепляем udev-правилом по признаку rotational
- ✕ **Пачка правок без baseline**
  20 параметров разом — эффект неатрибутируем; меняем один блок за итерацию и сверяем wrk/fio/pgbench с зафиксированным baseline
- ✕ **file-max поднят, LimitNOFILE — нет**
  Процесс упирается в лимит systemd, а не ядра; поднимаем цепочку fs.nr_open → DefaultLimitNOFILE → LimitNOFILE юнита

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Профили в /etc/sysctl.d/*.conf под git, применение только через sysctl --system
- somaxconn/tcp_max_syn_backlog под нагрузку + backlog в самом приложении
- vm.swappiness=1–10 и THP=never на серверах СУБД
- Baseline-замеры wrk/fio/iperf3 до любых правок

НОРМАЛЬНО

- BBR + fq на интернет-фронтах; TCP-буферы посчитаны по BDP канала
- Планировщик I/O udev-правилом: none — NVMe, mq-deadline — SSD, bfq — HDD
- dirty_background_ratio/dirty_ratio под профиль записи + LimitNOFILE в юнитах
- Мониторинг ListenOverflows, ретрансмитов и swap в Zabbix/Prometheus

ХОРОШО

- Статические hugepages под shared_buffers PostgreSQL, huge_pages=on
- NUMA-пиннинг СУБД через systemd NUMAPolicy=bind/NUMAMask
- Ansible-роль: профили по ролям серверов и суточный контроль дрейфа sysctl
- Регрессионные бенчмарки после каждого обновления ядра и ревизия профиля

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Все sysctl-правки живут в /etc/sysctl.d/ и в git, а не сделаны echo «на живую»?
- [ ] Есть baseline-замеры (wrk, fio, iperf3, pgbench) до тюнинга, чтобы доказать эффект каждой правки?
- [ ] somaxconn согласован с backlog в nginx/Redis/приложении, а не поднят в одиночку?
- [ ] THP переведён в never так, что это переживает перезагрузку (GRUB, а не echo в /sys)?
- [ ] Планировщик I/O задан udev-правилом под тип диска (NVMe/SSD/HDD)?
- [ ] Мониторинг видит ListenOverflows, TCP-ретрансмиты и активность swap?
- [ ] Лимиты дескрипторов подняты на всех уровнях: fs.nr_open, LimitNOFILE, само приложение?
- [ ] TCP-буферы посчитаны под полосу вашего канала, а не скопированы из чужого гайда?
- [ ] После обновления ядра/дистрибутива профиль перепроверяется — дефолты параметров меняются?
- [ ] Профиль восстанавливается автоматически (Ansible), если сервер пересобрали с нуля?

Если хотя бы на два вопроса ответ «нет» или «не знаю» — тема требует внимания.

ЧТО МЫ ДЕЛАЕМ

Как поможет ITFresh

ITFresh — ИТ-аутсорсинг для юридических лиц до 50 рабочих мест в Москве и области. 15+ лет практики, собственная инфраструктура в дата-центре МТС (8 серверов Dell Xeon Platinum).

- Аудит текущих sysctl и лимитов + baseline-бенчмарки нагрузки: сеть, диск, СУБД
- Разработка и внедрение sysctl-профилей под роль сервера: фронт, СУБД, файловый
- Настройка hugepages/NUMA для PostgreSQL и серверов 1С с проверкой pgbench до/после
- Мониторинг деградаций в Zabbix/Prometheus: очереди TCP, ретрансмиты, swap, latency диска
- Сопровождение: тираж профилей Ansible, контроль дрейфа и ревизия после обновлений ядра

15+
лет в ИТ-поддержке

50
рабочих мест — наш профиль

МТС
дата-центр, Москва

КОНТАКТЫ

Обсудить вашу задачу

Сайт	**itfresh.ru**
Телефон	**+7 903 729-62-41**
Telegram	**@ITfresh_Boss**

Бесплатно посмотрим вашу инфраструктуру по этому чек-листу и скажем, где тонко — без обязательств.

itfresh.ru

ДОКУМЕНТАЦИЯ И НАШ СТЕК

Техническая база

01 IP Sysctl (networking/ip-sysctl) (docs.kernel.org — 6.x)

02 Documentation for /proc/sys/vm (admin-guide/sysctl/vm) (docs.kernel.org — 6.x)

03 Documentation for /proc/sys/net (admin-guide/sysctl/net) (docs.kernel.org — 6.x)

04 Transparent Hugepage Support (admin-guide/mm/transhuge) (docs.kernel.org — 6.x)

05 HugeTLB Pages (admin-guide/mm/hugetlbpage) (docs.kernel.org — 6.x)

06 systemd.exec — LimitNOFILE, NUMAPolicy/NUMAMask (freedesktop.org — v257)

07 PostgreSQL: Managing Kernel Resources, huge_pages (postgresql.org — 17)

08 Наш шаблон профилей 9x-*.conf по ролям серверов (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.