Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Linux cgroups v2: лимиты CPU, памяти и I/O для сервисов

Как мы изолируем 1С, PostgreSQL, Docker и бэкапы на одном сервере через systemd и cgroups v2

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

На одном сервере клиента обычно соседствуют PostgreSQL, сервер 1С, nginx, Docker и ночной бэкап. Без лимитов один разросшийся процесс забирает всю память или диск, а OOM-killer ядра убивает не виновника, а самый крупный процесс — чаще всего базу. Мы раскладываем сервисы по slice'ам cgroups v2 с гарантиями и потолками, чтобы сбой одного не ронял остальные.

Почему это важно бизнесу

- OOM-kill PostgreSQL или rphost 1С посреди дня = откат транзакций, слетевшие сеансы бухгалтерии и 20–40 минут простоя на перезапуск
- Бэкап или пересборка индексов без IOWeight душат диск: 1С и сайт «висят» ночью и утром, пока задача не закончится
- Контейнер с утечкой памяти без MemoryMax уводит в swap весь хост — деградируют все сервисы вместо одного
- Учёт по cgroup даёт цифры для capacity planning: видно, кому реально нужен второй сервер, а кому — просто лимит
- Гарантии MemoryMin/CPUWeight позволяют плотнее размещать сервисы на одном сервере без покупки железа

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

v2

Единая иерархия /sys/fs/cgroup — дефолт Ubuntu 22.04+, Debian 12, RHEL 9; v1 больше не внедряем

cgroup-v2 admin-guide, kernel.org

252+

systemd ≥252 (Debian 12, RHEL 9; Ubuntu 24.04 — 255): CPUShares/MemoryLimit там уже deprecated

systemd.resource-control(5)

80/90 %

MemoryHigh 80% / MemoryMax 90% бюджета сервиса: throttling и сброс кэша идут раньше OOM-kill

наш стандарт

60 % / 30 с

Порог memory.pressure и выдержка, после которых systemd-oomd гасит cgroup — дефолты oomd.conf

oomd.conf(5), systemd v255

100 мс

Период CPUQuota по умолчанию (допустимо 1–1000 мс); интерактивным сервисам снижаем до 10–20 мс

CPUQuotaPeriodSec, systemd

1–10000

CPUWeight/IOWeight, дефолт 100: базе 1000, бэкапу 20 — приоритет при конкуренции, не квота

cpu.weight / io.weight, kernel.org

РЕАЛИЗАЦИЯ 1 ИЗ 3

Слайсы для 1С и PostgreSQL на одном сервере

Что настраиваем

Сервер приложений 1С 8.3 (ragent/rphost) и PostgreSQL 16 на Ubuntu 24.04, 64 ГБ RAM, 16 vCPU

Как мы это делаем

1. Проверяем режим: mount | grep cgroup2 и cat /sys/fs/cgroup/cgroup.controllers → cpu io memory pids; DefaultMemoryAccounting=yes в system.conf
2. Создаём db.slice и app.slice (drop-in в /etc/systemd/system/*.slice.d): PostgreSQL → Slice=db.slice, srv1cv8-8.3.25@default.service → Slice=app.slice
3. db.slice: MemoryMin=24G MemoryLow=32G MemoryHigh=44G MemoryMax=48G MemorySwapMax=0 CPUWeight=1000 IOWeight=800
4. postgresql@16-main: OOMScoreAdjust=-900 OOMPolicy=continue Environment=PG_OOM_ADJUST_FILE=/proc/self/oom_score_adj; app.slice: MemoryHigh=12G MemoryMax=14G
5. Применяем без рестарта: systemctl set-property db.slice MemoryMax=48G; контроль: systemctl show -p MemoryCurrent,EffectiveMemoryMax db.slice

РЕЗУЛЬТАТ

Тяжёлый отчёт в 1С упирается в лимит app.slice и тормозит только rphost — PostgreSQL сохраняет 24 ГБ гарантированной памяти и весь shared_buffers. OOM-kill базы посреди дня исключён: при исчерпании бюджета ядро выбирает жертву внутри app.slice, а не postmaster.

КЛЮЧЕВОЙ НЮАНС

shared_buffers — это shmem, он входит в memory.current cgroup базы. MemoryMax ниже shared_buffers + work_mem × max_connections даёт OOM под нагрузкой; бюджет считаем с запасом на page cache.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Docker-контейнеры и ночные задачи: потолки и приоритеты I/O

Что настраиваем

Хост с Docker Engine 28 (Bitrix/CRM в контейнерах), nginx на хосте, ночной бэкап restic + pg_dump на тот же NVMe

Как мы это делаем

1. Docker на cgroup v2 сам берёт cgroupdriver=systemd: контейнеры живут в system.slice/docker-<id>.scope; в compose задаём mem_limit, cpus, pids_limit
2. Общий потолок над всеми контейнерами: drop-in docker.service с MemoryHigh=20G MemoryMax=24G TasksMax=8192 CPUWeight=300; Delegate=yes оставляем
3. Бэкап — transient-юнит: systemd-run --unit=backup -p Slice=batch.slice -p IOWeight=20 -p CPUWeight=20 -p MemoryMax=4G restic backup ...
4. Полка для batch.slice на устройстве: IOReadBandwidthMax=/dev/nvme0n1 200M, IOWriteBandwidthMax=/dev/nvme0n1 100M; сверяем io.max и io.stat
5. Диагностика конкуренции: systemd-cgtop -d 2 --order=io и cat /sys/fs/cgroup/batch.slice/io.pressure — full avg10 > 20% = пересмотр лимита

РЕЗУЛЬТАТ

Ночной бэкап перестаёт «класть» сайт и 1С: при простое диска он забирает всю полосу, при конкуренции уступает по весу 20 против 100. Контейнер с утечкой упирается в свой memory.max и перезапускается по restart-политике, не утягивая хост в swap.

КЛЮЧЕВОЙ НЮАНС

io.max и io.weight работают на блочном устройстве: для LVM/mdraid указываем итоговый dm-/md-узел, иначе лимит молча не применится. Буферизованная запись режется только на ext4/XFS/btrfs с cgroup writeback.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Мониторинг cgroups: PSI, OOM-события и алерты в Zabbix

Что настраиваем

Linux-серверы клиентов под Zabbix 7.0 LTS (agent 2); на части стендов — Prometheus node_exporter + cAdvisor

Как мы это делаем

1. Шаблон «Systemd by Zabbix agent 2»: systemd.unit.discovery + systemd.unit.get — состояние юнитов; поверх — свои UserParameter по cgroup-файлам
2. UserParameter=cg.memevents[*]: awk по /sys/fs/cgroup/$1/memory.events (oom_kill); cg.pressure[*]: full avg60 из memory.pressure и cpu.pressure
3. Триггеры: дельта oom_kill > 0 → High; memory.pressure full avg60 > 10% 5 мин → Warning; рост nr_throttled в cpu.stat у latency-сервиса → ревизия CPUQuota
4. Prometheus-стенды: node_exporter --collector.pressure (PSI хоста) и --collector.systemd; per-cgroup память и CPU — cAdvisor по system.slice
5. systemd-oomd: ManagedOOMMemoryPressure=kill на batch.slice и user.slice, ManagedOOMPreference=omit на db.slice; проверка — oomctl dump

РЕЗУЛЬТАТ

Об OOM-kill и throttling мы узнаём из мониторинга, а не из звонка бухгалтера: memory.events и PSI показывают, кто именно упёрся в лимит и сколько времени сервис ждал ресурс. Лимиты пересматриваем по данным, а не по ощущениям.

КЛЮЧЕВОЙ НЮАНС

top/htop дают мгновенный снимок, а cgroup — накопленные счётчики: memory.peak, throttled_usec, total в PSI. Для алертов берём дельты, а не абсолюты, иначе первый OOM-kill после ребута теряется.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **OOMPolicy=stop гасит весь сервис**
  После OOM-kill одного воркера systemd по умолчанию (DefaultOOMPolicy=stop) останавливает юнит целиком. Для PostgreSQL/1С ставим OOMPolicy=continue.

- ✕ **Лимит памяти без swap: зависание, не kill**
  memory.high без swap уводит cgroup в затяжной reclaim, а SwapUsedLimit в systemd-oomd без swap не работает. Оставляем swap 4–8 ГБ и MemorySwapMax.

- ✕ **CPUQuota режет латентность**
  Квота выдаёт рывки: сервис исчерпал долю за 100 мс и ждёт. Смотрим nr_throttled в cpu.stat; для 1С/веба — CPUWeight, квоту только батчам.

- ✕ **Процессы в промежуточном cgroup**
  Правило «no internal processes»: нельзя держать процессы в cgroup с включённым subtree_control. Ручные группы — только листья или systemd-run --scope.

- ✕ **Ручной echo в /sys/fs/cgroup не живёт**
  systemd пересоздаёт дерево при старте и затирает ручные лимиты. Всё — через drop-in или set-property (пишет в /etc/systemd/system.control/).

- ✕ **Delegate=yes у Docker и podman**
  В делегированный subtree systemd не лезет: лимиты контейнерам задаём через Docker (--memory, --cpus), а на docker.service вешаем общий потолок.

- ✕ **Page cache считается в memory.current**
  Файловый кэш бэкапа или pg_dump наполняет memory.current. С MemoryHigh ядро сбросит кэш до kill; без него MemoryMax срабатывает раньше времени.

- ✕ **Не то устройство в io.max (LVM/md)**
  IOReadBandwidthMax=/dev/sda на LVM-томе не действует. Указываем /dev/mapper/... или /dev/md0 и сверяем major:minor в io.max.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Проверить cgroup v2: cgroup.controllers = cpu io memory pids, учёт памяти включён
- MemoryHigh/MemoryMax + OOMPolicy=continue на каждый критичный сервис (БД, 1С, веб)
- Бэкапы и разовые задачи только через systemd-run с IOWeight=20 и MemoryMax
- Еженедельная проверка memory.events (oom_kill) по всем сервисам

НОРМАЛЬНО

- Слайсы db/app/web/batch с MemoryMin/Low и CPUWeight, сервисы привязаны через Slice=
- Docker: общий потолок на docker.service, лимиты mem_limit/cpus/pids_limit в compose
- Zabbix: UserParameter по memory.events, memory.pressure, cpu.stat с триггерами
- systemd-oomd: kill на batch/user.slice, omit на db.slice, swap оставлен

ХОРОШО

- IODeviceLatencyTargetSec для БД (25 мс) и io.max на batch — защита латентности
- PSI-алерты (full avg60) и дельты oom_kill в мониторинге, ревью лимитов раз в квартал
- AllowedCPUs для БД и 1С на многоядерных хостах, CPUQuotaPeriodSec=10ms для веба
- Лимиты в конфиг-менеджменте (Ansible), тест на стенде через systemd-run перед прод

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Сервер на cgroup v2 (cgroup2 в mount, файл cgroup.controllers есть), а не в гибридном v1-режиме?
- [ ] У БД и сервера 1С заданы MemoryMin/MemoryHigh/MemoryMax и OOMPolicy=continue?
- [ ] Есть отдельный batch.slice для бэкапов с IOWeight и MemoryMax, задачи идут через systemd-run?
- [ ] На хосте есть swap и MemorySwapMax по сервисам, а не выключенный swap «для скорости»?
- [ ] Docker работает с cgroupdriver=systemd и у docker.service есть общий потолок памяти и TasksMax?
- [ ] Мониторинг снимает memory.events (oom_kill), memory.pressure и cpu.stat с триггерами?
- [ ] Устройства в IO-лимитах — реальные блочные узлы (dm-/md/nvme), проверено по io.max?
- [ ] Лимиты хранятся в drop-in /etc/systemd и в конфиг-менеджменте, а не в ручных echo?
- [ ] systemd-oomd настроен осознанно: kill на batch/user, omit на db, проверено oomctl dump?
- [ ] Есть бюджет памяти сервера: сумма MemoryMin ≤ RAM, сумма MemoryMax с запасом ядру и кэшу?

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 сервера: кто сколько ест по systemd-cgtop и memory.events, карта OOM-kill за месяц, бюджет памяти и CPU
- Проектирование слайсов и лимитов под 1С, PostgreSQL, Docker, веб и бэкапы; внедрение через drop-in без простоя
- Мониторинг cgroups в Zabbix 7.0 / Prometheus: PSI, OOM-события, throttling, алерты в Telegram
- Настройка systemd-oomd и OOM-политик, чтобы при нехватке памяти гибли батчи, а не база
- Сопровождение: квартальный пересмотр лимитов по фактическим данным, перенос настроек в Ansible

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 Control Group v2 — admin-guide (memory/cpu/io controllers, PSI) (docs.kernel.org — 6.x)

02 systemd.resource-control(5) — CPUWeight, MemoryHigh/Max, IO*, ManagedOOM* (freedesktop.org — v255)

03 systemd-oomd(8), oomd.conf(5), oomctl(1) (freedesktop.org — v255)

04 systemd-run(1), systemd.slice(5), systemd-cgtop(1) (freedesktop.org — v255)

05 RHEL 9: Managing, monitoring and updating the kernel — cgroups-v2 (docs.redhat.com — RHEL 9)

06 Docker Engine: Runtime metrics, Resource constraints (cgroup v2) (docs.docker.com — 28.x)

07 PostgreSQL 16: Managing Kernel Resources — Linux Memory Overcommit (postgresql.org — 16)

08 Шаблон Systemd by Zabbix agent 2 + наш UserParameter cg.* (zabbix.com — 7.0 LTS)

Основано на официальной документации продуктов и нашей практике внедрения.