Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Корпоративный сайт тормозит: Nginx или всё-таки админ

Наша методика: разделяем вину фронта, PHP-FPM и БД по $upstream_response_time

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Клиент видит «502 Bad Gateway» и по надписи в браузере обвиняет Nginx. Мы приходим на VPS, где нет ни одного числа: access.log в дефолтном combined без таймингов, stub_status не включён, slowlog выключен. Спор «сервер или сайт» тянется неделями, каталог отдаётся по 8 секунд, форма заявки падает в пик. Мы сначала инструментуем стек, потом чиним — и точно знаем, чей это слой.

Почему это важно бизнесу

- Медленный каталог и 502 в пик = брошенные корзины и заявки, которые не дошли до менеджера
- Пока причина не локализована, платятся часы и хостеру, и разработчику — за диагностику одного и того же
- Лежащий сайт перед отправкой КП или в тендерный день бьёт по репутации сильнее, чем стоит починка
- Реклама ведёт на страницу, которая открывается 8 секунд: бюджет откручивается, а заявок нет
- Без метрик деградация замечается по звонку клиента, а не по алерту — реакция вместо профилактики

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

1.30.4

Стабильная ветка nginx, на которую мы переводим фронты; mainline 1.31.x — только на стендах

nginx.org, ветка 1.30

800 мс

Наш порог TTFB: выше — копаем сервер и бэкенд, ниже — правим фронтенд и картинки

наш стандарт

1000

Дефолт keepalive_requests начиная с 1.19.10 (было 100); к upstream отдельно задаём keepalive

ngx_http_core_module

10 мин

Дефолт inactive в proxy_cache_path; под каталог ставим 24h, иначе кэш вымывается ночью

ngx_http_proxy_module

5 с

Наш request_slowlog_timeout в пуле PHP-FPM (дефолт 0 = выкл): дольше — стек в slowlog

php.net, PHP-FPM 8.3

60 с

Интервал съёма stub_status и pm.status_path в мониторинг — базовая линия Active/Waiting

наш стандарт

РЕАЛИЗАЦИЯ 1 ИЗ 3

Инструментовка: сначала цифры, потом тюнинг

Что настраиваем

Фронт nginx 1.30 из репозитория nginx.org на VPS клиента (Ubuntu 24.04) + пул PHP-FPM 8.3 и MySQL за ним

Как мы это делаем

1. Свой log_format timing: добавляем $request_time, $upstream_response_time, $upstream_connect_time, $upstream_cache_status; logrotate — 14 суток
2. Включаем stub_status на 127.0.0.1/nginx_status и pm.status_path = /fpm-status в пуле, снимаем агентом мониторинга раз в 60 с
3. PHP-FPM: slowlog = /var/log/php-fpm/www.slow.log, request_slowlog_timeout = 5s, request_terminate_timeout = 120s — получаем стек виновного скрипта
4. БД: в MySQL slow_query_log = ON и long_query_time = 1, в PostgreSQL log_min_duration_statement = 1000ms; отдельно смотрим попадание в буферный пул
5. Считаем дельту $request_time минус $upstream_response_time: устойчивое расхождение — это сеть и медленный клиент, а не бэкенд

РЕЗУЛЬТАТ

За один рабочий день получаем распределение времени ответа по слоям: сколько съел nginx, сколько PHP-FPM, сколько запрос к БД. Дальше правим только тот слой, где реально время, и заказчик видит замер до и после, а не обещания.

КЛЮЧЕВОЙ НЮАНС

Дефолтный combined-формат лога не содержит ни одного тайминга. Пока в access.log нет $upstream_response_time, спор «сервер или сайт» не решается в принципе.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Тюнинг nginx: то, за что он действительно отвечает

Что настраиваем

Фронт-конфиг узла: раздача статики, сжатие, кэш проксирования, TLS-хендшейк, лимиты запросов и соединений

Как мы это делаем

1. worker_processes auto, worker_rlimit_nofile 65535, events { worker_connections 8192 }; параллельно поднимаем LimitNOFILE в systemd-юните
2. Статика: sendfile on, tcp_nopush on, open_file_cache max=10000 inactive=60s, expires 1y и add_header Cache-Control "public, immutable" для ассетов с хэшем
3. HTTP/2 включаем директивой http2 on (параметр listen ... http2 объявлен устаревшим в 1.25.1), ssl_session_cache shared:SSL:10m, ssl_session_timeout 1d
4. proxy_cache_path levels=1:2 keys_zone=site:64m inactive=24h max_size=8g; proxy_cache_lock on, proxy_cache_use_stale updating error timeout
5. proxy_cache_background_update on плюс limit_req_zone на форму входа и админку — пик не проваливает весь сайт целиком

РЕЗУЛЬТАТ

Статика уходит из PHP и отдаётся ядром, повторные запросы каталога закрываются из кэша, а обновление горячей страницы не превращается в лавину одинаковых запросов к бэкенду. Скачок посещаемости перестаёт быть аварией.

КЛЮЧЕВОЙ НЮАНС

proxy_cache молча выключается, если бэкенд шлёт Set-Cookie или Cache-Control: no-store. Проверяем не конфиг, а $upstream_cache_status в логе: HIT / MISS / BYPASS.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Бэкенд и лимиты: откуда берутся 502 и 504

Что настраиваем

Связка nginx → PHP-FPM 8.3 через unix-сокет → MySQL/PostgreSQL на том же VPS клиента

Как мы это делаем

1. pm.max_children считаем по фактическому RSS воркера: (RAM минус БД минус ОС) делим на средний RSS; pm = dynamic, pm.max_requests = 500 против утечек
2. Согласуем listen.backlog пула (дефолт 511 на Linux) с net.core.somaxconn — иначе в пик получаем 502 при формально живом PHP
3. fastcgi_read_timeout и proxy_read_timeout поднимаем точечно в location выгрузок и отчётов 1С, а не глобально на весь сервер
4. emergency_restart_threshold = 10 и emergency_restart_interval = 1m в php-fpm.conf ловят серию аварийных выходов воркеров вместо тихой деградации
5. Разводим диагнозы по error.log: 502 — не удалось соединиться с upstream, 504 — upstream не ответил в отведённое время

РЕЗУЛЬТАТ

Пул перестаёт быть чёрным ящиком: видно, упирается ли сайт в число воркеров, в память или в БД. Пиковые 502 уходят, а длинные операции 1С живут в своём location и не тянут за собой таймауты остального сайта.

КЛЮЧЕВОЙ НЮАНС

Глобально задранный read_timeout не лечит, а прячет: воркеры копятся, и вместо честной ошибки клиент получает минуту ожидания и затем 504 всё равно.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **worker_connections задрали до сотни тысяч**
  Без worker_rlimit_nofile и LimitNOFILE в юните упрёмся в лимит дескрипторов: в error.log посыплется too many open files, а RPS не вырастет.

- ✕ **gzip накрутили на уже сжатое**
  Сжатие jpeg, png и архивов жжёт CPU без выигрыша. Задаём gzip_types явно, gzip_comp_level 5 и gzip_min_length 1024 вместо дефолтных 20 байт.

- ✕ **Глобальный proxy_read_timeout 600s**
  Маскирует медленный бэкенд и копит воркеры до лавины 504. Длинные таймауты держим только в location выгрузок и отчётов.

- ✕ **Кэш включили, заголовки не проверили**
  Set-Cookie и no-store от бэкенда отключают кэширование молча. Контролируем по $upstream_cache_status, а не по наличию директив.

- ✕ **HTTP/2 через параметр listen**
  Параметр listen ... http2 устарел с 1.25.1, нужна директива http2 on. Иначе HTTP/2 тихо не включается в части server-блоков.

- ✕ **pm.max_children по формуле из интернета**
  Считаем по реальному RSS воркера конкретного сайта. Иначе в пик срабатывает OOM-killer, гасит БД, и мы получаем 502 на ровном месте.

- ✕ **Правка конфига без nginx -t**
  Любое изменение: nginx -t, затем reload. Конфиги фронта храним в git на нашей стороне — видно, кто и когда внёс правку.

- ✕ **Тюнинг раньше измерения**
  Крутить директивы, не зная долю nginx во времени ответа, — лотерея. Сначала тайминги в логе и статус-страницы, потом любые изменения.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- log_format с $request_time и $upstream_response_time, ротация логов 14 суток
- stub_status на 127.0.0.1 и pm.status_path в пуле PHP-FPM
- gzip с явным gzip_types, expires для статики, nginx -t перед каждым reload
- Раздельный error.log по сайтам и алерт на всплеск 5xx

НОРМАЛЬНО

- request_slowlog_timeout 5s в пуле и long_query_time 1 в MySQL
- pm.max_children посчитан по фактическому RSS, pm.max_requests против утечек
- proxy_cache или fastcgi_cache с cache_lock on и use_stale updating
- http2 on, ssl_session_cache shared:SSL:10m, ssl_session_timeout 1d

ХОРОШО

- Метрики nginx и PHP-FPM в Zabbix/Grafana, дашборд по слоям и TTFB
- limit_req на формы и админку, limit_conn на тяжёлые разделы
- Конфиги фронта в git, выкатка через шаблон и проверку nginx -t в CI
- Нагрузочный прогон профиля пика перед акцией и релизом

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] В access.log есть $request_time и $upstream_response_time по каждому сайту?
- [ ] Включены stub_status и pm.status_path пула PHP-FPM, метрики снимаются раз в минуту?
- [ ] Заданы request_slowlog_timeout в пуле и slow_query_log с long_query_time в БД?
- [ ] Знаете ли вы фактический RSS воркера и откуда взято текущее pm.max_children?
- [ ] Статика отдаётся nginx с expires и open_file_cache, а не через PHP?
- [ ] Кэширование проверено по $upstream_cache_status в логе, а не по наличию директив?
- [ ] HTTP/2 включён директивой http2 on и подтверждён ответом curl --http2 -I?
- [ ] Длинные таймауты вынесены в отдельный location для выгрузок 1С и отчётов?
- [ ] Есть ли алерт на рост доли 5xx и на превышение порога TTFB 800 мс?
- [ ] Конфиги nginx версионируются, и каждый reload предваряется nginx -t?

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).

- Инструментуем стек за один визит: тайминги в логах, stub_status, статус пула, slowlog БД
- Локализуем причину по слоям и даём отчёт с замерами до и после, а не общие рекомендации
- Приводим фронт-конфиг к нашему шаблону: кэш, сжатие, статика, TLS, лимиты запросов
- Пересчитываем пул PHP-FPM под реальную память узла и убираем пиковые 502 и 504
- Ставим мониторинг с алертами на 5xx и TTFB и берём фронт на регулярное обслуживание

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 nginx: ngx_http_core_module (keepalive, sendfile, open_file_cache) (nginx.org — 1.30)

02 nginx: ngx_http_proxy_module (proxy_cache_path, use_stale) (nginx.org — 1.30)

03 nginx: ngx_http_v2_module, директива http2 (nginx.org — 1.30)

04 nginx: ngx_http_log_module и переменные модуля upstream (nginx.org — 1.30)

05 nginx: ngx_http_limit_req_module и ngx_http_stub_status_module (nginx.org — 1.30)

06 PHP-FPM: конфигурация пулов (pm, slowlog, emergency_restart) (php.net — 8.3)

07 MySQL: журнал медленных запросов (slow_query_log) (dev.mysql.com — 8.0)

08 Наш шаблон фронт-конфига и чек-лист диагностики TTFB (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.