Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Valkey вместо Redis: переводим кэш и очереди без даунтайма

Наша методика: аудит инстансов Redis, ввод Valkey 8.1/9.1, миграция репликацией, HA и монитори...

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

У клиентов Redis держит кэш сайта, сессии 1С-Битрикс и очереди фоновых задач. После смены лицензии ветка Redis 7.2 доживает без развития, а новые версии несут лицензионные ограничения при размещении у провайдеров. Мы переводим инстансы на Valkey: тот же протокол и форматы RDB/AOF, свободная BSD-3-Clause и прирост пропускной способности за счёт io-threads — без правки кода прил...

Почему это важно бизнесу

- Падение кэша гонит всю нагрузку в БД: сайт и 1С-интеграции замедляются в разы, в пик — простой продаж
- Лицензии SSPLv1/RSALv2 — юридический риск при размещении в облаке и передаче сервиса подрядчику
- Ветка без патчей безопасности плюс открытый порт 6379 — типовая точка входа, сканеры находят её за часы
- Миграция репликацией переносит данные фоном: окно переключения — секунды, код приложений не меняется

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

8.1.x

Рабочая ветка Valkey для продакшена: стабильные патч-релизы; 9.1 закладываем на новые проекты

valkey.io, релизы 2026

7.2.4

Точка форка: полная совместимость с Redis 7.2 по протоколу, командам и формату RDB/AOF

по докам valkey.io

io-threads 6

I/O-потоки на 8-ядерном узле (на 4 ядрах — 2–3); по умолчанию 1 — включаем при упоре сетевого…

valkey.conf 8.1

everysec

appendfsync для AOF: fsync раз в секунду — при сбое питания теряется максимум 1 секунда записи

по докам valkey.io

75%

maxmemory от RAM узла + политика вытеснения: запас на copy-on-write при BGSAVE и полной синхро…

наш стандарт

16384

Хэш-слоты Valkey Cluster: полное покрытие проверяем командой valkey-cli --cluster check

cluster tutorial, valkey.io

РЕАЛИЗАЦИЯ 1 ИЗ 3

Аудит Redis и подготовка узла Valkey

Что настраиваем

Standalone-инстансы Redis 5–7.2 на Linux-узлах клиента: кэш сайта, сессии, очереди фоновых задач

Как мы это делаем

1. Снимаем INFO server/clients/memory: версия, пиковый used_memory, клиентские библиотеки, наличие модулей (MODULE LIST)
2. Проверяем совместимость: модули RedisJSON/RediSearch/TimeSeries с Valkey не работают — замену (valkey-json, valkey-search, valkey-bloom) подбираем до миграции
3. Ставим Valkey из репозитория дистрибутива (Debian 13 — ветка 8.1, Debian 12 — backports) или официального Docker-образа valkey/valkey; сервис valkey-server.service
4. Переносим конфиг: redis.conf → /etc/valkey/valkey.conf без смены формата; сверяем maxmemory, appendonly, requirepass
5. Закрываем периметр: bind на внутренний интерфейс, protected-mode yes, пароль/ACL, порт 6379 только из LAN

РЕЗУЛЬТАТ

Узел готов к переключению заранее: конфиг перенесён один-в-один, периметр закрыт, известны точный объём данных и список клиентов — само переключение перестаёт быть разведкой боем.

КЛЮЧЕВОЙ НЮАНС

Valkey читает redis.conf без правок, но мы сверяем параметры по самодокументированному valkey.conf текущей ветки — это и есть основная конфигурационная дока продукта.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Миграция через репликацию: zero-downtime

Что настраиваем

Боевой Redis → Valkey на соседнем узле или VM, приложения переключаются по DNS/VIP

Как мы это делаем

1. Поднимаем Valkey репликой боевого Redis: replicaof <master> 6379 и masterauth в valkey.conf
2. Ждём полной синхронизации: INFO replication → master_link_status:up и master_sync_in_progress:0
3. Промотируем реплику командой REPLICAOF NO ONE и переводим приложения по DNS/VIP — окно в секунды
4. Сверяем DBSIZE и контрольные ключи на обоих узлах до останова старого Redis
5. Redis останавливаем, но его RDB-снапшот храним 7 дней как точку отката

РЕЗУЛЬТАТ

Данные переезжают фоном при работающем сервисе; клиентские библиотеки (redis-py, ioredis, go-redis) работают без изменений — меняется только адрес. Откат — обратное переключение на ещё живой Redis.

КЛЮЧЕВОЙ НЮАНС

Признак готовности — только связка master_link_status:up + master_sync_in_progress:0. На больших базах заранее поднимаем repl-backlog-size, чтобы обрыв линка не вызвал полный resync.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Эксплуатация: отказоустойчивость и мониторинг

Что настраиваем

Промышленный контур: мастер + реплика, наблюдение в Zabbix/Prometheus

Как мы это делаем

1. HA по размеру задачи: реплика + Sentinel (кворум 2 из 3) для failover или Cluster 3 master + 3 replica для шардирования
2. Метрики в мониторинг: used_memory, hit rate (keyspace_hits/misses), connected_clients, rdb_last_bgsave_status
3. Алерты: рост evicted_keys, master_link_status:down, latency по valkey-cli --latency выше порога 1 мс
4. Бэкап: BGSAVE по расписанию + ночная выгрузка dump.rdb в бэкап-хранилище отдельным заданием
5. io-threads включаем по факту замера: valkey-benchmark до и после, число потоков меньше числа физических ядер

РЕЗУЛЬТАТ

Деградацию видим до жалоб пользователей: падение hit rate и вытеснение ключей ловятся алертами, отказ мастера отрабатывается автоматически, восстановление из dump.rdb отрепетировано.

КЛЮЧЕВОЙ НЮАНС

Sentinel и Cluster — разные инструменты: Sentinel даёт failover без шардирования, Cluster делит данные по 16384 слотам. Для баз до 10–20 ГБ обычно достаточно связки реплика + Sentinel.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **io-threads не по числу ядер**
  Потоки конкурируют с основным потоком за CPU, latency растёт. По valkey.conf: 6 потоков на 8 ядрах, 2–3 на 4; пользу подтверждаем valkey-benchmark

- ✕ **Модули Redis Ltd в проде**
  RedisJSON/RediSearch/TimeSeries с Valkey несовместимы. Проверяем MODULE LIST на аудите и переходим на valkey-json/valkey-search до переключения

- ✕ **maxmemory не задан**
  Инстанс растёт до OOM-killer и роняет соседние сервисы. Задаём maxmemory ~75% RAM и политику: allkeys-lru для кэша, noeviction для очередей

- ✕ **AOF выключен на очередях**
  Кэш терять можно, очереди и сессии — нет. Для них appendonly yes + appendfsync everysec; для чистого кэша достаточно RDB-снапшотов

- ✕ **Порт 6379 открыт наружу**
  Сканеры находят открытый инстанс за часы. bind на LAN-интерфейс, requirepass или ACL-пользователи, за пределами сегмента — только tls-port

- ✕ **Переключение до конца синка**
  REPLICAOF NO ONE при master_sync_in_progress:1 теряет данные. Ждём 0, сверяем DBSIZE и контрольные ключи на мастере и реплике

- ✕ **repl-backlog по умолчанию**
  На объёмных базах короткий обрыв линка приводит к полному resync и просадке мастера. Поднимаем repl-backlog-size под интенсивность записи

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- Valkey 8.1.x из репозитория дистрибутива или официального Docker-образа вместо дожив…
- maxmemory + политика вытеснения, requirepass, bind на внутренний интерфейс
- Миграция репликацией с контролем master_sync_in_progress:0
- RDB-снапшоты и ночная выгрузка dump.rdb в бэкап-хранилище

НОРМАЛЬНО

- io-threads по рекомендации valkey.conf с замером valkey-benchmark до/после
- AOF everysec для очередей и сессий, RDB — для кэша
- Мониторинг: hit rate, used_memory, evicted_keys, статус BGSAVE
- ACL-пользователи под каждое приложение вместо общего пароля

ХОРОШО

- Реплика + Sentinel (кворум 2/3) или Cluster 3M+3R при росте данных
- TLS (tls-port) на клиентов и репликацию за пределами сегмента
- Отрепетированный откат: восстановление из снапшота с зафиксированным RTO
- Плановые апгрейды по веткам valkey.io без перескока мажорных версий

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Известно ли, какие приложения ходят в Redis/Valkey и какими клиентскими библиотеками?
- [ ] Проверен ли MODULE LIST — нет ли модулей Redis Ltd без плана замены?
- [ ] Задан ли maxmemory и политика вытеснения под роль инстанса (кэш/очередь)?
- [ ] Включён ли AOF там, где данные терять нельзя (очереди, сессии)?
- [ ] Закрыт ли порт 6379 от интернета, настроен ли пароль или ACL?
- [ ] Есть ли реплика и проверенный сценарий переключения мастера?
- [ ] Снимаются ли метрики hit rate, памяти, evicted_keys и настроены ли алерты?
- [ ] Выгружается ли dump.rdb в бэкап и проверялось ли восстановление из него?
- [ ] Зафиксировано ли, на какой ветке Valkey живёт прод и когда следующий апгрейд?

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 текущего Redis: версии, модули, клиенты, лицензионные и security-риски
- Миграция на Valkey без даунтайма через репликацию, с планом и точкой отката
- Настройка HA (Sentinel или Cluster) и тюнинг io-threads/памяти под нагрузку
- Мониторинг Zabbix/Prometheus с алертами и передача кэш-слоя на нашу поддержку

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 Releases and versioning schema (valkey.io — 2026)

02 Installation (valkey.io — 2026)

03 Самодокументированный конфиг valkey.conf (valkey.io — 8.1)

04 Replication (valkey.io — 8.1)

05 Persistence (RDB/AOF) (valkey.io — 8.1)

06 Cluster tutorial (valkey.io — 8.1)

07 Security / ACL (valkey.io — 8.1)

08 Наш чек-лист миграции кэш-слоя + Zabbix-шаблон (itfresh.ru — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.