Ай-ТИ Фреш

ТЕХНИЧЕСКИЙ РАЗБОР

Огромные архивы клиента: ratarmount вместо распаковки

Как мы достаём один файл из tar.gz на сотни гигабайт за минуты — FUSE-монтирование с SQLite-ин…

Июль 2026

itfresh.ru · ИТ-аутсорсинг для юридических лиц

О ЧЁМ ЭТОТ РАЗБОР

Суть проблемы

Недельные бэкапы файловых серверов и 1С у клиентов — tar.gz на 300–500 ГБ на удалённом FTP/SFTP. Запрос «нужен один договор за прошлый сентябрь» превращается в 2,5–3 часа: скачать архив целиком, найти 800+ ГБ под распаковку, ждать gzip. Пока идёт распаковка, юрист не сдаёт документы, бухгалтерия не закрывает период, а инженер занят одной задачей вместо пяти.

Почему это важно бизнесу

- Точечное восстановление занимает часы инженера — это прямые деньги по договору обслуживания
- Распаковка требует в 2 раза больше диска, чем архив: временные 800+ ГБ на боевом сервере — риск переполнения
- Срочные документы (договоры, акты, сканы) нужны «сегодня», а не после ночной распаковки
- Контроль: администратор видит содержимое бэкапа как папку и может проверить его без восстановления
- Меньше трафика: по сети идут только запрошенные байты, а не весь архив

ЧТО МЫ ЗАКЛАДЫВАЕМ

Ключевые параметры реализации

1.3.0

Версия ratarmount (май 2026), которую ставим через pipx с extras [full] — включает rapidgzip,...

ratarmount 1.3.0, pypi.org

16 МиБ

--gzip-seek-point-spacing по умолчанию: шаг точек seek по несжатым данным; каждая точка ≈ 32 K...

README ratarmount 1.3.0, раздел Usage

<1 с

Повторное монтирование при готовом .index.sqlite — ищется рядом с архивом, затем в ~/.ratarmou...

README ratarmount 1.3.0, раздел Index

146 МБ

Индекс для архива 124 ГБ с 1,26 млн файлов: размер зависит от числа записей, а не от объёма —...

README ratarmount 1.3.0, пример индекса

-P 0

--parallelization по умолчанию 0 = все ядра при первом проходе rapidgzip; на боевом сервере ог...

README ratarmount 1.3.0, опция -P

ssh://

Штатные схемы доступа через fsspec: ssh://, sftp://, ftp://, https://, s3://, smb:// — без ска...

README ratarmount 1.3.0, раздел Remote files

РЕАЛИЗАЦИЯ 1 ИЗ 3

Стенд восстановления: ratarmount поверх SFTP-хранилища бэкапов

Что настраиваем

Linux-стенд инженера (Ubuntu 24.04) или служебная VM рядом с FTP-хранилищем Veeam; архивы tar.gz на SFTP

Как мы это делаем

1. apt install pipx fuse3; pipx install 'ratarmount[full]'; проверяем ratarmount --version — в выводе должны быть rapidgzip и indexed_zstd
2. Учётка на SFTP только на чтение (chroot на каталог клиента); ключ ed25519, пароли в сейфе — в URL не пишем
3. Первый монт: ratarmount -P 4 --index-file /var/cache/ratarmount/<клиент>_<дата>.index.sqlite sftp://user@host/backup/2025-09/x.tar.gz /mnt/restore
4. Проверка: ls -la /mnt/restore, du -sh индекса, sha256sum извлечённого файла сверяем с эталоном на сервере-источнике
5. fusermount3 -u /mnt/restore после копирования; индекс оставляем — следующий запрос по тому же архиву стартует за секунду

РЕЗУЛЬТАТ

Один файл из 400-гигабайтного архива при готовом индексе — за 2–12 минут вместо 3 часов; на стенде нужно место только под индекс (сотни МБ), сеть загружается на объём файла. Без индекса первый проход читает архив целиком — по 1 Гбит около часа.

КЛЮЧЕВОЙ НЮАНС

Индекс ищется сначала рядом с архивом, затем в ~/.ratarmount — на read-only SFTP первое место недоступно, поэтому всегда задаём --index-file явно, иначе индекс теряется между стендами и строится заново.

РЕАЛИЗАЦИЯ 2 ИЗ 3

Регламент бэкапа: делаем архивы «ratarmount-friendly» на стороне клиента

Что настраиваем

Скрипты ночного бэкапа файлового сервера и выгрузок 1С (.dt, srvinfo) на Linux-хосте клиента

Как мы это делаем

1. Индекс строим на источнике сразу после tar: ratarmount --recreate-index --index-file x.tar.gz.index.sqlite x.tar.gz /tmp/idx; fusermount3 -u; индекс кладём рядом с…
2. Новые архивы — многофреймовый zstd: tar -cf - dir | t2sz -l 3 -s 16M -o x.tar.zst; indexed_zstd делает seek только к границам фреймов, tar --zstd даёт один фрейм
3. Крупные бинарные файлы (db.dt, VHDX) — отдельный tar без сжатия: seek внутри несжатого tar мгновенный
4. Для срезов (bash/rsync-скрипты) — только полные tar; GNU incremental монтируем с --detect-gnu-incremental, но не полагаемся на него
5. Sanity-check после бэкапа: ratarmount --verify-mtime + ls 3 контрольных путей + sha256 одного файла — результат в лог и в Zabbix

РЕЗУЛЬТАТ

Индекс едет вместе с архивом на FTP: при восстановлении инженер пропускает первый проход по архиву целиком и сразу открывает нужный каталог. Проверка целостности бэкапа становится ежедневной, а не «когда понадобилось».

КЛЮЧЕВОЙ НЮАНС

Автопоиск индекса привязан к пути архива, а совпадение проверяется по размеру (по mtime — только с --verify-mtime): при копировании на другое хранилище передаём индекс через --index-file.

РЕАЛИЗАЦИЯ 3 ИЗ 3

Эксплуатация и безопасность точки монтирования

Что настраиваем

Служебные VM восстановления, доступ инженеров ITfresh и администратора клиента

Как мы это делаем

1. Монтируем только под непривилегированным пользователем; -o allow_other не включаем, user_allow_other в /etc/fuse.conf не трогаем
2. Точка монта в /mnt/restore/<клиент>/<дата>; systemd-unit Type=simple с ratarmount -f, ExecStop=fusermount3 -u, TimeoutStopSec=30
3. Кэш индексов /var/cache/ratarmount на SSD, ротация по tmpreaper (30 дней); индексы бэкапов 1С — хранение до конца квартала
4. Для 7z/zip/rar внутри tar — --recursive; для архивов с паролем — --password-file, файл 0600 в каталоге инженера
5. После работ: fusermount3 -u, проверка mount | grep ratarmount, запись в журнал восстановлений с sha256 выданного файла

РЕЗУЛЬТАТ

Точки монтирования не «зависают» после ухода инженера, индексы не расползаются по домашним каталогам, а каждый выданный из бэкапа файл имеет контрольную сумму и запись в журнале — это закрывает вопросы аудита клиента.

КЛЮЧЕВОЙ НЮАНС

FUSE-монт живёт в сессии пользователя: при обрыве SSH ratarmount без -f и systemd может остаться с оборванным транспортом — поэтому долгие операции только через unit или tmux.

ЧТО МЫ ВЫУЧИЛИ НА ПРАКТИКЕ

Подводные камни

- ✕ **Индекс строится при каждом монте**
  На read-only SFTP ratarmount не может записать .index.sqlite рядом с архивом, а ~/.ratarmount на каждом стенде свой; задаём --index-file явно.

- ✕ **Слишком частые точки seek в gzip**
  Каждая точка seek ≈ 32 КиБ в индексе; при малом --gzip-seek-point-spacing на архиве 400 ГБ индекс разрастается. Оставляем 16 МиБ по умолчанию, для zs…

- ✕ **Все ядра боевого сервера под rapidgzip**
  По умолчанию -P 0 использует все ядра; на хосте с 1С это тормозит rphost. Ограничиваем -P 2..4 и запускаем через nice/ionice.

- ✕ **Пароль в URL ssh://user:pass@**
  URL попадает в history и ps. Используем ключи и ~/.ssh/config, пароль к архиву — только --password-file с правами 0600.

- ✕ **Забытая точка монтирования**
  После обрыва SSH FUSE-монт остаётся с мёртвым транспортом и ls зависает. Долгие сессии — только под systemd/tmux, всегда fusermount3 -u.

- ✕ **Индекс не совпал с архивом**
  Архив пересобран с тем же именем и размером — по умолчанию сверяется только размер, старый индекс даёт мусорные offset'ы. Включаем --verify-mtime или…

- ✕ **Вложенные архивы не видны**
  tar внутри tar или zip с выгрузкой 1С монтируются как файлы. Для дерева включаем --recursive, но помним про рост времени индексации.

- ✕ **Ожидание скорости диска по сети**
  Первый проход ограничен каналом до FTP: 400 ГБ по 1 Гбит — около часа. Поэтому индекс строим на источнике и переносим вместе с архивом.

УРОВНИ РЕАЛИЗАЦИИ

Как правильно

МИНИМУМ

- ratarmount 1.3.x через pipx с [full] на стенде инженера
- Явный --index-file на локальном SSD для каждого архива
- Монт по sftp:// без скачивания архива, только read-only учётка
- fusermount3 -u и проверка mount после каждого восстановления

НОРМАЛЬНО

- Индекс создаётся на источнике сразу после бэкапа и хранится рядом с архивом
- Новые архивы — многофреймовый zstd (t2sz), крупные бинарники (.dt) — без сжатия отде…
- Кэш индексов /var/cache/ratarmount с ротацией 30 дней
- Ежедневный sanity-check бэкапа: монт + ls контрольных путей в Zabbix

ХОРОШО

- Служебная VM восстановления рядом с хранилищем: монт по локальной сети, а не по инте…
- systemd-unit на долгие сессии, -P ограничен, nice/ionice для индексации
- Журнал восстановлений с sha256 каждого выданного файла
- Регламент клиенту: запрос файла из бэкапа — до 30 минут по SLA

ПРОВЕРЬТЕ У СЕБЯ

Чек-лист самопроверки

- [ ] Установлен ratarmount 1.3.x с rapidgzip и indexed_zstd (ratarmount --version показывает бэкенды)?
- [ ] Для каждого архива задан --index-file на локальном SSD, а не пересоздаётся при каждом монте?
- [ ] Индекс строится на источнике бэкапа и уезжает на FTP вместе с архивом?
- [ ] Учётка на SFTP/FTP для восстановления — только чтение и chroot на каталог клиента?
- [ ] Пароли и ключи не фигурируют в URL, history и ps (ключи ed25519, --password-file 0600)?
- [ ] На боевых хостах ограничен -P и включён nice/ionice при построении индекса?
- [ ] Новые бэкапы делаются в многофреймовом zstd (t2sz/pzstd) или без сжатия для крупных бинарных файлов?
- [ ] Есть ежедневный тест: монт архива + ls контрольных путей + алерт в Zabbix при ошибке?
- [ ] После восстановления точка монтирования снята (fusermount3 -u) и это проверяется?
- [ ] Каждый файл, выданный из бэкапа, записан в журнал с sha256 и датой?

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).

- Разворачиваем стенд восстановления с ratarmount и доступом к вашему хранилищу бэкапов по SFTP
- Перестраиваем регламент бэкапов: многофреймовый zstd, индексы на источнике, проверка монтированием каждую ночь
- Подключаем контроль бэкапов в Zabbix: успешность архива, размер, тест монтирования
- Точечное восстановление файлов и выгрузок 1С из архивов любого объёма по SLA до 30 минут
- Обучаем администратора клиента: памятка на 1 страницу и systemd-unit под ключ

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 ratarmount — README: Usage, Index, Remote files, zstd (github.com/mxmlnkn/ratarmount — 1.3.0 (2…)

02 ratarmount — PyPI: extras [full], [fsspec]; fsspec-схемы ssh/sftp/ftp/s3 (pypi.org — 1.3.0 (2…)

03 rapidgzip — параллельная распаковка gzip и seek-индекс (github.com/mxmlnkn/rapidgzip — 0.16.0 (…)

04 indexed_zstd и t2sz — random access по фреймам zstd, сборка многофреймовых… (github.com/martinellimarco — 2026)

05 libfuse 3 — fusermount3, /etc/fuse.conf, allow_other (github.com/libfuse/libfuse — 3.x)

06 GNU tar — --zstd, -T0, incremental (--listed-incremental) (gnu.org/software/tar — 1.35 (20…)

07 Шаблон ITfresh: restore-mount.service + скрипт индексации бэкапа (наш шаблон — 2026)

Основано на официальной документации продуктов и нашей практике внедрения.