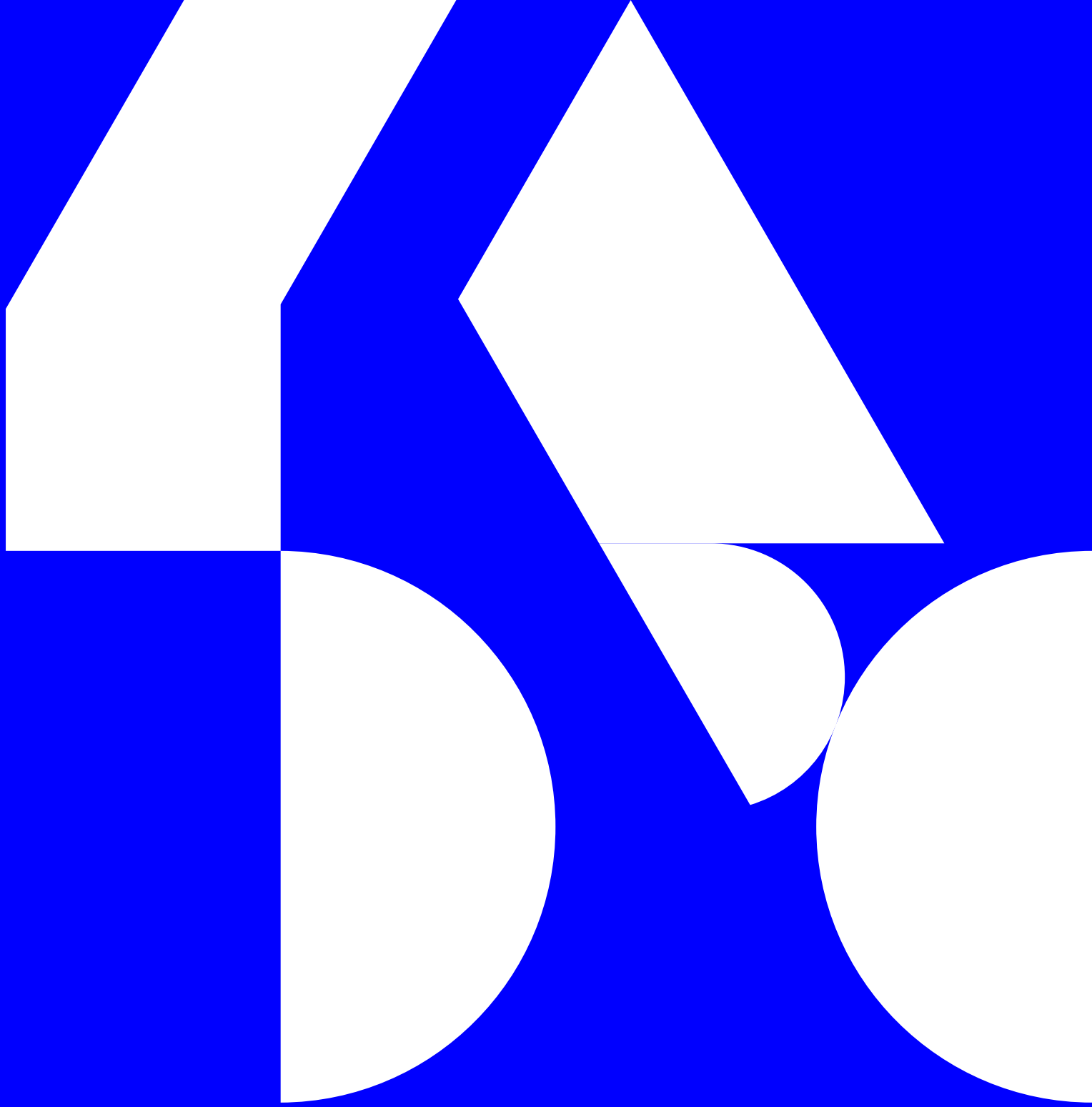


WHITE PAPER

ОСОБЕННОСТИ ЭФФЕКТИВНОГО ЧТЕНИЯ, ЗАПИСИ И ХРАНЕНИЯ ДАННЫХ В TATLIN.OBJECT

О WHITE PAPER

Данный документ описывает архитектурные принципы и подходы, а также рекомендации по их применению при построении корпоративных решений на основе продуктов производства компании YADRO.

Документ предназначен для широкой аудитории, но также может включать материалы, требующие специальной профессиональной подготовки, ориентированные на системных архитекторов, администраторов, сервисных инженеров и других ИТ-специалистов, участвующих в планировании, проектировании и внедрении комплексных решений.

Представленная информация основана на результатах исследований, а также экспертизе специалистов YADRO. Документ носит не предписывающий, а информационный характер, целью которого является информирование получателей документа о предмете произведенного исследования и о его результатах. При этом конкретные результаты и эффективность конечных решений зависят от множества факторов, которые могут различаться в зависимости от особенностей инфраструктуры и условий эксплуатации. Ни при каких обстоятельствах YADRO не несет ответственности за последствия применения получателем документа рекомендаций, инструкций, подходов и решений, указанных в документе.

Запрещается копировать, воспроизводить, изменять, публиковать, передавать, распространять, продавать информацию (часть информации), содержащуюся в документе, или создавать производные работы на основе содержания документа без предварительного письменного разрешения YADRO.

Любая информация или утверждения относительно возможностей, емкости, производительности или других характеристик продукции, указанной в документе, предоставляются исключительно на условиях «как есть» и приведена исключительно в целях описания произведенного исследования. YADRO оставляет за собой право вносить изменения/обновлять информацию в документе без уведомления.

«YADRO» является зарегистрированным товарным знаком ООО «КНС ГРУПП». Другие названия продуктов и компаний, упомянутые здесь, могут быть товарными знаками или торговыми наименованиями их соответствующих владельцев.

Актуальные версии документации публикуются на информационном ресурсе YADRO по адресу service.yadro.com.

ОГЛАВЛЕНИЕ

1. ВВЕДЕНИЕ 4
2. МНОГОПОТОЧНЫЙ ДОСТУП 5
3. АРХИТЕКТУРА УЗЛА: ШАРДЫ И ИСПОЛЬЗОВАНИЕ SSD 6
   - 3.1. Структура SSD-раздела шарда 6
   - 3.2. Управление пространством SSD 6
   - 3.3. Write Cache 7
   - 3.4. Метабаза 7
4. РЕЖИМЫ РАБОТЫ ШАРДА 8
   - 4.1. READ-WRITE (Чтение и запись) 8
   - 4.2. READ-ONLY (Только чтение) 8
   - 4.3. DEGRADED-READ-ONLY (Деградированный режим чтения) 8
5. ХРАНЕНИЕ ИНФОРМАЦИИ НА УЗЛАХ СХД 9
   - 5.1. Хранение объектов малого размера (< 128 КиБ) 9
   - 5.2. Хранение объектов среднего размера (128 КиБ – 64 МиБ) 9
   - 5.3. Хранение объектов большого размера (> 64 МиБ) 10
6. ХРАНЕНИЕ ОБЪЕКТОВ БОЛЬШОГО РАЗМЕРА: ПРОТОКОЛ S3 И ИСПОЛЬЗОВАНИЕ MPU 12
7. НАТИВНЫЙ ПРОТОКОЛ FROSTFS (СЛОЙ ХРАНЕНИЯ) И ЕГО ОТОБРАЖЕНИЕ В S3 14
   - 7.1. Общая информация 14
   - 7.2. Уникальная функциональность frostfs, отсутствующая в S3 15
   - 7.3. Запуск frostfs-cli без интерактивного ввода пароля 16
   - 7.4. Особенности создания контейнеров FrostFS в TATLIN.OBJECT 1.6 16
   - 7.5. Ограничение доступа к данным — APE Rule 17
   - 7.6. Особенности работы с контейнером FrostFS в именованном неймспейсе 17
   - 7.7. Таймауты при создании контейнеров/бакетов 18
   - 7.8. Обновление карты сети и восстановление данных отключенного узла 18
   - 7.9. Выбор узла для доступа к контейнеру/бакету 18
8. РАБОТА С FROSTFS-SDK 20
9. ОСОБЕННОСТИ РАБОТЫ С S3 21
   - 9.1. Управление доступом через IAM Policy 21
   - 9.2. Оптимизация удаления данных в S3 21
   - 9.3. ETag S3-объекта 21
   - 9.4. Листинг бакета с большим числом объектов 22
   - 9.5. Совместимость со сторонними S3 клиентами 23
   - 9.6. Использование curl для работы с S3 23
10. УСКОРЕНИЕ ОДНОПОТОЧНЫХ ЗАГРУЗОК ФАЙЛОВ ПО ССЫЛКАМ PRESIGNED-URL 24
11. ОБЩИЕ РЕКОМЕНДАЦИИ ДЛЯ ОПТИМИЗАЦИИ ОПЕРАЦИЙ ЗАПИСИ/ЧТЕНИЯ 25

ВВЕДЕНИЕ

TATLIN.OBJECT — это программно-аппаратный комплекс, реализующий объектное хранилище данных с поддержкой протоколов S3, HTTPS и FrostFS. В основе системы лежит эффективный децентрализованный распределенный механизм хранения данных, спроектированный для работы в нестабильной сетевой среде, а также в условиях отказа накопителей и узлов хранения.

Управление системой децентрализовано и может осуществляться с любого узла СХД. Каждый узел хранения использует все доступное локальное дисковое пространство для размещения данных и их индексации. Избыточность и отказоустойчивость обеспечиваются на уровне всего кластера. Часть узлов выполняет функции мониторинга состояния кластера и поддержания актуального реестра узлов в дополнение к хранению данных. Эта информация хранится в специализированной реплицируемой базе данных, копии которой распределены по узлам. Таким образом, в архитектуре системы отсутствуют единые точки отказа и центральные компоненты, ограничивающие ее производительность. Это позволяет размещать узлы в различных центрах обработки данных (ЦОД) и географически удаленных регионах без значительного увеличения операционных затрат.

Настоящий документ описывает архитектурные подходы к хранению пользовательских данных в системе хранения данных (СХД), особенности работы с объектными протоколами с использованием инструментов CLI и SDK, а также аспекты совместимости со сторонним программным обеспечением.

Документ предназначен для разработчиков, архитекторов, ИТ-специалистов и администраторов, отвечающих за проектирование, внедрение и эксплуатацию решений на основе объектного хранилища данных TATLIN.OBJECT.

ВЕРСИИ ИСПОЛЬЗУЕМОГО ПО	
TATLIN.OBJECT	1.6.x — 1.8

МНОГОПОТОЧНЫЙ ДОСТУП

Объектные СХД предназначены для одновременной обработки большого количества запросов на чтение и запись. Для достижения максимальной производительности операции чтения и записи должны выполняться параллельно в большом количестве потоков, направленных на один или несколько узлов системы.

Ограничение: Максимальное количество одновременных соединений на фронтенде одного узла составляет 2000.
Рекомендация: Оптимальное количество потоков для операций записи и чтения зависит от размера объектов и должно подбираться индивидуально для каждой рабочей нагрузки.

Пример: Запись S3 (Конфигурация: TATLIN.OBJECT версии 1.7, 4 узла, политика хранения REP2)

РАЗМЕР ОБЪЕКТА	ПОТОКОВ НА УЗЕЛ	СУММАРНАЯ СКОРОСТЬ	ПРОПУСКНАЯ СПОСОБНОСТЬ (ОБ/С)
8КиБ	800	54 МиБ/с	~6631
1МиБ	700	1,3 ГиБ/с	~1194
128МиБ	200	1,8 ГиБ/с	~13,33

Пример: Чтение S3 (Конфигурация: TATLIN.OBJECT версии 1.7, 4 узла, политика хранения REP2)

РАЗМЕР ОБЪЕКТА	ПОТОКОВ НА УЗЕЛ	СУММАРНАЯ СКОРОСТЬ	ПРОПУСКНАЯ СПОСОБНОСТЬ (ОБ/С)
8КиБ	800	82 МиБ/с	~9998
1МиБ	700	2,35 ГиБ/с	~2165
128МиБ	200	3,2 ГиБ/с	~23,9

Особенности многопоточной записи

Многопоточная запись подразумевает одновременную загрузку нескольких объектов на узлы системы. Для ускорения загрузки одного крупного объекта следует использовать механизм многочастной загрузки (MPU — Multi Part Upload). Фрагменты (чанки) объекта могут загружаться параллельно[1]. Операции по разделению объекта на фрагменты, их сборке на стороне системы хранения и контролю целостности выполняются средствами AWS S3 SDK.

Особенности многопоточного чтения

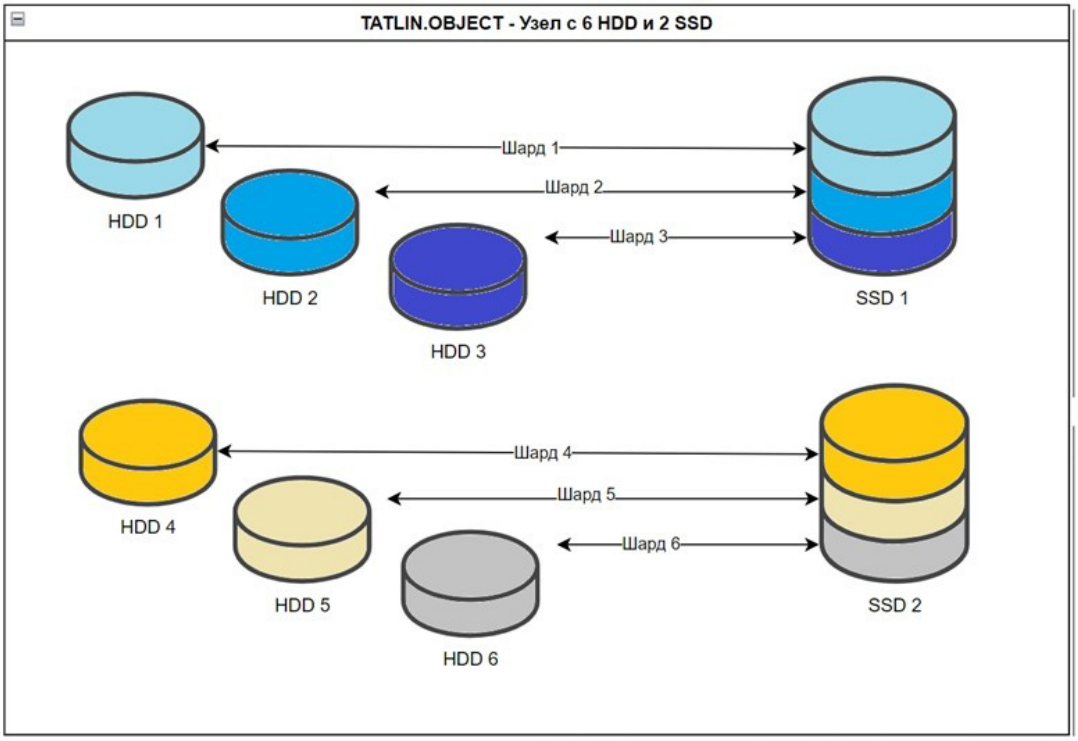
Многопоточное чтение подразумевает одновременное скачивание нескольких объектов.

- **Скачивание целиком**
  Стандартные средства AWS S3 SDK скачивают один объект в рамках одного потока. Один объект может скачиваться одновременно несколькими клиентами.
- **Скачивание частями**
  Протокол S3 поддерживает скачивание части объекта (запрос Range для указания диапазона байт или PartNumber для запроса конкретного фрагмента многочастного объекта)[2]. Это позволяет разным клиентам параллельно скачивать разные части одного объекта.
  Важно: Задача объединения скачанных частей и проверки целостности конечного объекта не входит функционал AWS S3 SDK и должна быть реализована на стороне клиентского приложения.

АРХИТЕКТУРА УЗЛА: ШАРДЫ И ИСПОЛЬЗОВАНИЕ SSD

Данные на узле СХД организованы в логические единицы — шарды. Каждый шард использует ресурсы как HDD, так и SSD. Один шард соответствует одному HDD-диску большого объема в узле и включает в себя выделенный раздел на SSD[3].

Таким образом, общее количество шардов на узле равно количеству установленных в нем HDD-дисков.



Структура SSD-раздела шарда

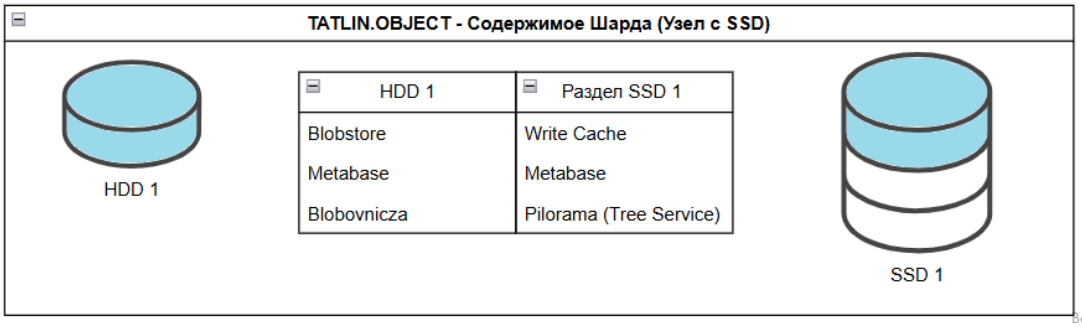
SSD-раздел каждого шарда разделен на три функциональные области:

1. SSD Write Cache — кэш для временного хранения записываемых объектов размером до 2 МиБ[4].
2. Метабаза — индекс всех объектов, хранящихся на данном шарде.
3. База данных CRDT-дерева - база, в которой хранится лог операций над деревом объектов..

Управление пространством SSD

Под SSD Write Cache одного шарда резервируется 256 ГиБ. Оставшееся пространство SSD (например, ~0.278 ТБ на шард для узла с 12 HDD) выделяется для метабазы и базы данных объектного сервиса. Эти области имеют вытесняющий приоритет. Это означает, что при росте объема метаданных сверх выделенного лимита размер SSD Write Cache будет автоматически уменьшаться.

Наличие SSD-раздела является ключевым фактором высокой производительности операций поиска и листинга объектов, особенно для протокола S3, интенсивно работающего с метаданными.



Если узел не оснащен SSD-разделами для шардов, все данные (как пользовательские, так и служебные) записываются и хранятся непосредственно на HDD.

Write Cache

Назначение: кэширование операций записи для объектов небольшого размера[5].

Принцип работы:

1. Объекты размером менее 2 МиБ[6] сначала записываются на SSD, а затем асинхронно переносятся на HDD.
2. Объекты размером более 2 МиБ записываются напрямую на HDD, минуя кэш.
3. Таким образом, SSD Write Cache ускоряет операции записи небольших объектов.
4. В операциях чтения кэш участвует только в том случае, если запрошенный объект еще не был выгружен на HDD (например, при чтении сразу после записи).

Метабаза

Назначение: индекс всех объектов шарда для ускорения операций поиска и листинга.

Принцип работы:

1. Обновляется при каждой операции записи или удаления объекта.
2. При конфигурации шарда с SSD метабаза размещается на SSD-диске для максимальной производительности.
3. При конфигурации без SSD метабаза размещается на HDD-диске.
4. В случае повреждения (например, после замены диска) метабаза автоматически реконструируется на основе данных, хранящихся на шарде.

РЕЖИМЫ РАБОТЫ ШАРДА

Любой статус шарда, отличный от READ-WRITE, READ-ONLY или DEGRADED-READ-ONLY, указывает на полную недоступность данных.

В этом случае система автоматически инициирует процедуру восстановления данных с других узлов кластера в соответствии с заданной политикой избыточности.

READ-WRITE (Чтение и запись)

Назначение: штатный режим работы с максимальной производительностью.

Принцип работы:

- Все сервисы системы используют шард для операций чтения и записи.
- Метабаза и база данных объектного сервиса (Пилорама) размещаются на SSD (при его наличии).
- Данные доступны для записи и чтения.
- Производительность соответствует максимальным показателям системы.

READ-ONLY (Только чтение)

Назначение: режим восстановления после ошибок ввода-вывода.

Принцип работы:

- Все сервисы системы используют шард исключительно для операций чтения.
- Метабаза и база данных объектного сервиса считываются с SSD (при его наличии).
- Запись новых данных невозможна.
- Производительность на чтение остается высокой.
- Условие активации режима работы: при достижении порогового значения ошибок чтения/записи.
- Действие при активации: требуется обратиться в сервисную службу YADRO.

DEGRADED-READ-ONLY (Деградированный режим чтения)

Назначение: аварийный режим при отказе SSD.

Принцип работы:

- Сервисы системы используют только HDD-часть шарда для чтения.
- SSD-раздел не используется.
- Метабаза и база данных объектного сервиса считываются с HDD (при их наличии на этом носителе).
- Производительность чтения объектов снижена до среднего уровня.
- Производительность операций листинга и поиска объектов низкая.
- Условие активации режима работы: при сбое в работе SSD-раздела.
- Действие при активации: требуется обратиться в сервисную службу YADRO.

ХРАНЕНИЕ ИНФОРМАЦИИ НА УЗЛАХ СХД

Система оптимизирует запись и хранение объектов в зависимости от их размера.

Хранение объектов малого размера (< 128 КиБ)

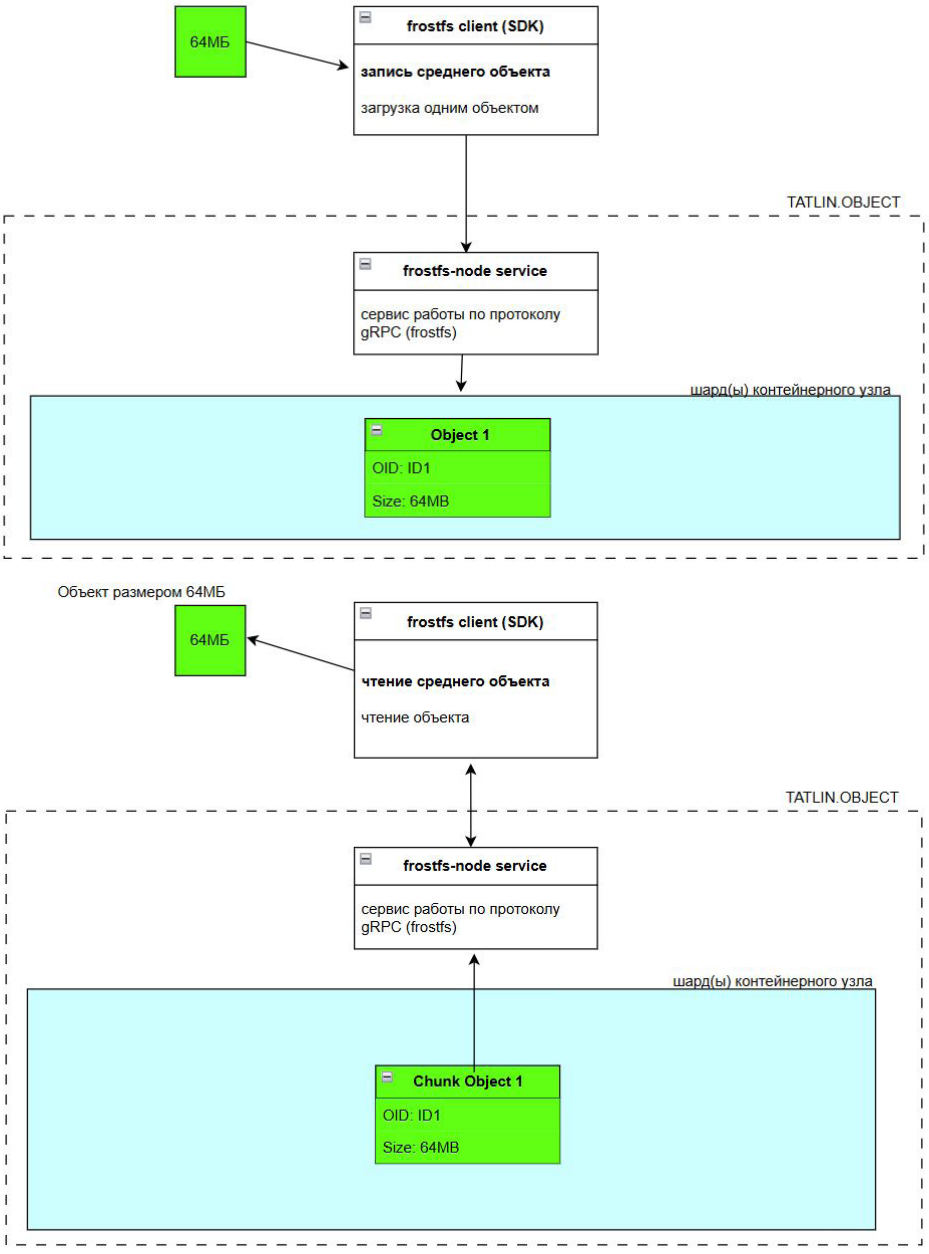
Принцип работы:

- Объекты размером менее 128 КиБ группируются и упаковываются в специализированные файлы баз данных на шарде — блобовницы.
- Запись оптимизирована за счет использования SSD Write Cache для первичного приема данных.
- В фоновом режиме данные переносятся и упаковываются в блобовницы на HDD.
- Один файл блобовницы занимает до 40 МиБ и записывается непосредственно на HDD-диск.

Преимущество: эффективное использование дискового пространства и снижение нагрузки на ввод-вывод за счет группировки множества мелких операций в одну крупную.

Хранение объектов среднего размера (128 КиБ – 64 МиБ)

Принцип работы: объекты данного диапазона сохраняются на шарде как отдельные независимые объекты.



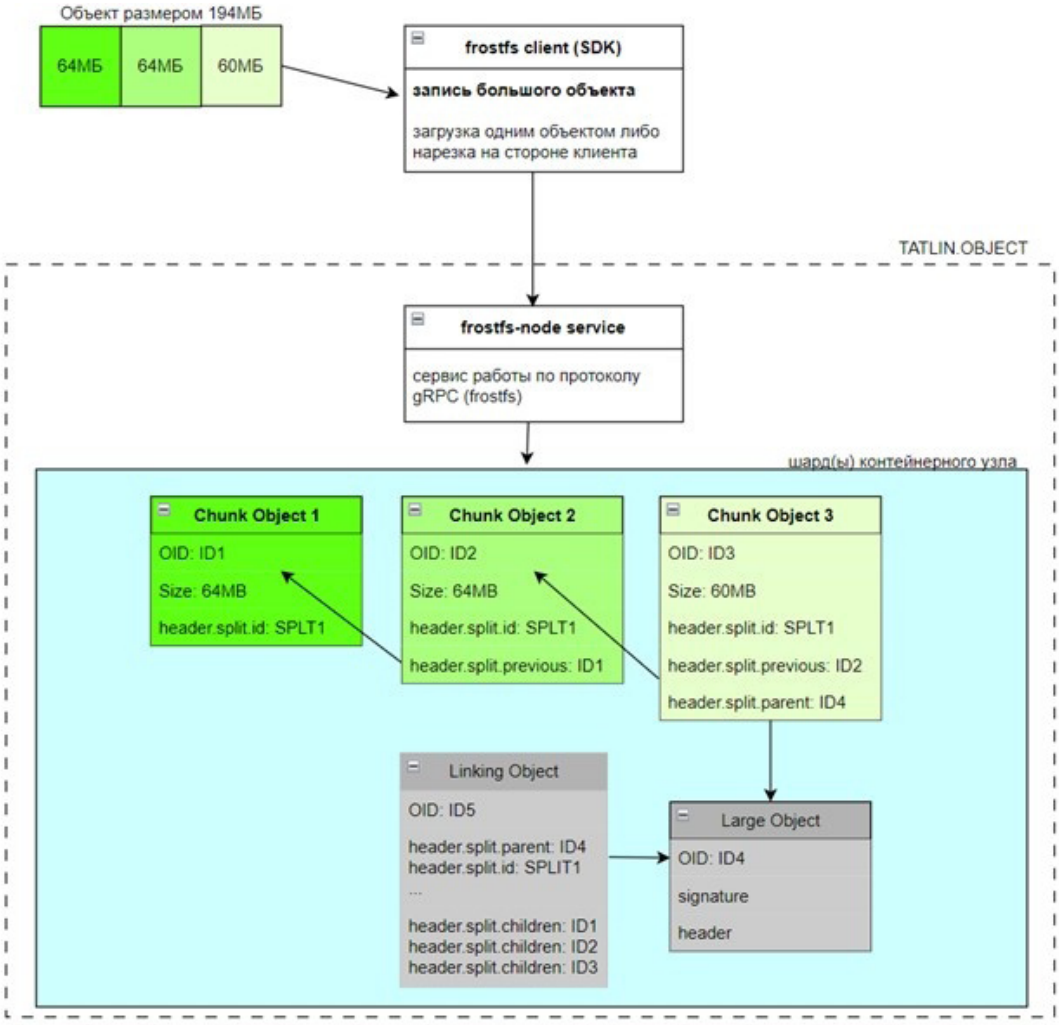
Хранение объектов большого размера (> 64 МиБ)

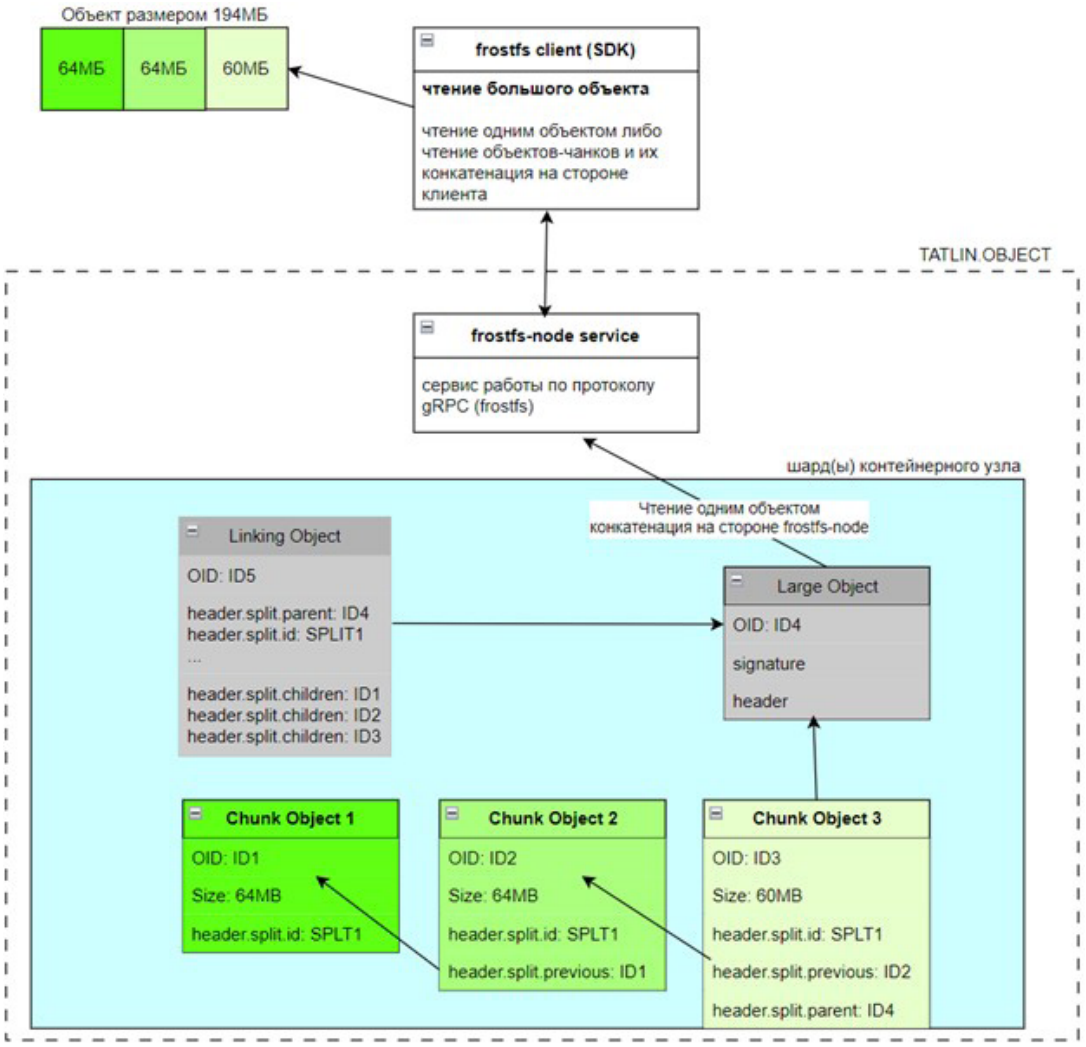
Принцип работы:

- При записи объект, превышающий 64 МиБ, автоматически разделяется системой хранения на части (чанки) фиксированного размера 64 МиБ[7].
- На уровне хранения большой объект представлен набором этих чанков.
- Каждый чанк содержит идентификатор (OID) предыдущего чанка, формируя цепочку.
- После завершения записи автоматически создаются два служебных объекта:
  - Linking Object: содержит упорядоченный список OID всех чанков. Используется для произвольного доступа (чтения данных из середины объекта).
  - Large Object: содержит OID последнего чанка. Используется для последовательного (потокового) чтения объекта от начала до конца.

Чтение большого объекта:

- Стандартное чтение (через Large Object) выполняется последовательно в один поток. Система автоматически объединяет (конкатенирует) чанки во время передачи данных, предоставляя клиенту объект как единое целое.
- Допускается прямое скачивание отдельных чанков.
  Важно: в этом случае задача сборки исходного объекта из частей ложится на клиентское приложение и не входит в функционал FrostFS SDK.




Объект размером 194МБ
64МБ
64МБ
60МБ
frostfs client (SDK)
чтение большого объекта
чтение одним объектом либо чтение объектов-чанков и их конкатенация на стороне клиента
TATLIN.OBJECT
frostfs-node service
сервис работы по протоколу gRPC (frostfs)
шард(ы) контейнерного узла
Чтение одним объектом конкатенация на стороне frostfs-node
Linking Object
OID: ID5
header.split.parent: ID4
header.split.id: SPLIT1
...
header.split.children: ID1
header.split.children: ID2
header.split.children: ID3
Large Object
OID: ID4
signature
header
Chunk Object 1
OID: ID1
Size: 64MB
header.split.id: SPLT1
Chunk Object 2
OID: ID2
Size: 64MB
header.split.id: SPLT1
header.split.previous: ID1
Chunk Object 3
OID: ID3
Size: 60MB
header.split.id: SPLT1
header.split.previous: ID2
header.split.parent: ID4

ХРАНЕНИЕ ОБЪЕКТОВ БОЛЬШОГО РАЗМЕРА: ПРОТОКОЛ S3 И ИСПОЛЬЗОВАНИЕ MPU

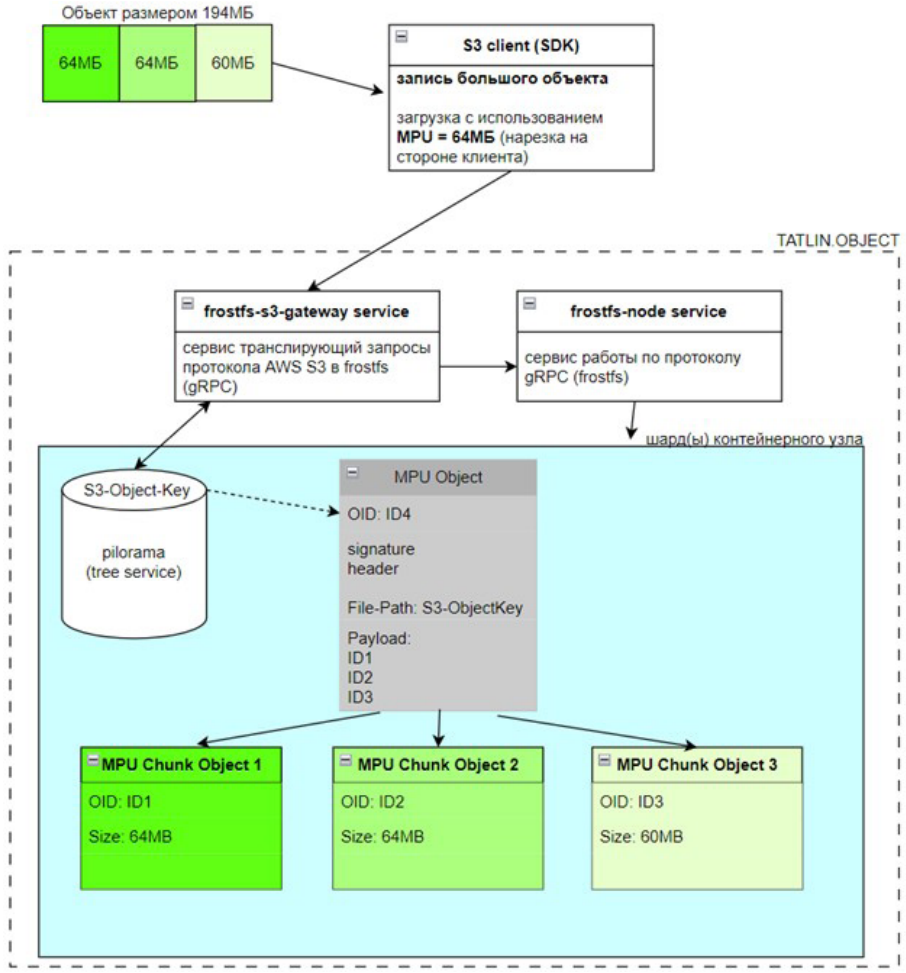
Загрузка крупных файлов (от 5 ГиБ[8]) единым объектом может приводить к таймаутам соединения[9]. Для таких объектов рекомендуется использовать механизм многочастной загрузки (MPU — Multi Part Upload)[10], который предполагает предварительное разделение объекта на части на стороне клиента.

Принцип работы MPU:

1. **Настройка**
   В клиентском приложении задается размер части (чанка) MPU.
2. **Загрузка**
   Чанки объекта загружаются на систему хранения последовательно или параллельно (для увеличения скорости).
3. **Формирование объекта**
   После загрузки всех чанков отправляется команда CompleteMultipartUpload с указанием списка загруженных частей.
4. **Хранение**
   На уровне хранения FrostFS объект сохраняется как набор отдельных чанков и служебный объект, содержащий упорядоченный список их идентификаторов (OID).

Рекомендации по выбору размера чанка MPU:

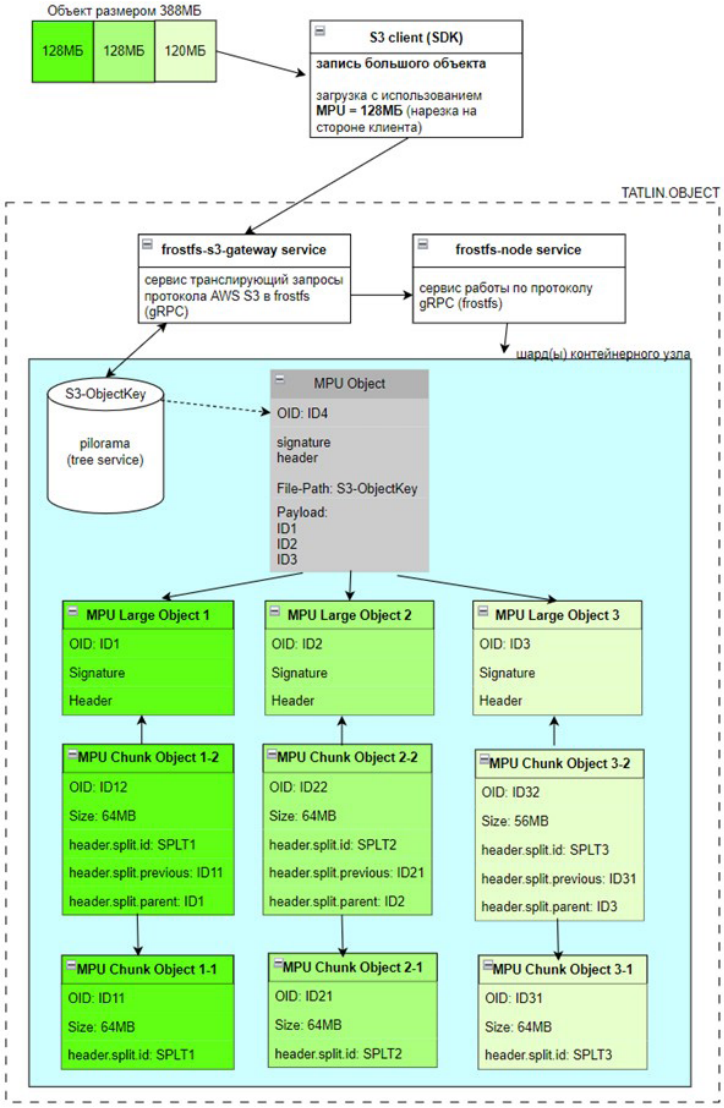
- Для объектов до 640 ГиБ рекомендуется использовать чанки размером 5–64 МиБ, чтобы общее количество частей не превышало 10 000[11].
  - 5 МиБ — минимальный размер чанка, определенный стандартом AWS S3.
  - 64 МиБ — максимальный размер, при превышении которого система автоматически разделит чанк на части по 64 МиБ[12].



- Для объектов свыше 640 ГиБ размер чанка должен быть более 64 МиБ при условии, что общее количество частей остается меньше 10 000[13]. Рекомендуется выбирать размер, кратный 64 МиБ.

РАЗМЕР ЧАНКА	МАКСИМАЛЬНЫЙ РАЗМЕР ОБЪЕКТА
128 МиБ	до 1280 ГиБ
192 МиБ	до 1920 ГиБ
256 МиБ	до 2560 ГиБ
320 МиБ	до 3200 ГиБ
384 МиБ	до 3840 ГиБ
448 МиБ	до 4480 ГиБ
512 МиБ	до 5120 ГиБ

Спецификация Amazon S3 поддерживает объекты размером до 5 ТБ. TATLIN.OBJECT допускает работу с более крупными объектами (с учетом ограничения в 5 ГиБ на чанк), однако рекомендуется придерживаться стандартных лимитов.



НАТИВНЫЙ ПРОТОКОЛ FROSTFS (СЛОЙ ХРАНЕНИЯ) И ЕГО ОТОБРАЖЕНИЕ В S3

Общая информация

TATLIN.OBJECT поддерживает два объектных протокола: нативный FrostFS и Amazon S3.
Слой хранения реализован на основе протокола FrostFS. Некоторые концепции S3 транслируются непосредственно на уровень FrostFS.

СООТВЕТСТВИЕ КОНЦЕПЦИЙ FROSTFS И S3

FrostFS	S3
Пользователь (UID) Wallet file / Wallet Password	IAM User (ARN) Key/Secret
Политика хранения	Location Constraint
Объект (OID) Атрибут объекта `FilePath`	Объект (Object Key)
Контейнер (CID) Атрибут контейнера `Name`	Бакет (Bucket Name)
rpc-endpoint (Базовый URL FrostFS, порт 8080)	S3 эндпоинт (Базовый URL S3, порт 443)
Namespace	Namespace
APE Policy	Bucket Policy

Расширение функциональности через сервисный слой

Для хранения дополнительных концепций S3 используется сервисный слой и его база данных (Пилорама), которые расширяют функциональность FrostFS:

- **Изменяемые атрибуты**
  Иммутабельные (неизменные) атрибуты объектов/контейнеров FrostFS дублируются в БД сервисного слоя для обеспечения возможности изменения через S3.
- **Дополнительные атрибуты**
  Добавление атрибутов и свойств S3, отсутствующих в FrostFS.
- **Конфигурации бакетов**
  Хранение S3 CORS, WEB Configuration, Bucket Policy, Lifecycle Policy.
- **Версионирование**
  Версионированные объекты S3 представляются как набор отдельных объектов FrostFS, связанных через сервисный слой.

Особенности взаимодействия протоколов

- К любому объекту в S3 бакете, созданному через S3 API, можно обратиться с использованием FrostFS API.
- Обратное не поддерживается: объекты, созданные через FrostFS API, недоступны через S3 API[14].
- FrostFS показывает на 10-20% более высокую производительность при операциях чтения/записи, так как не использует сервисный слой.
- Операции листинга и поиска в FrostFS работают быстрее и точнее благодаря отсутствию задержек синхронизации баз данных сервисного слоя.

Особенности работы с объектами

Объекты FrostFS иммутабельны. OID объекта вычисляется как хэш от хранимых данных и его атрибутов.
При изменении объекта создается новый объект с новым OID, что может приводить к снижению производительности при агрессивной перезаписи крупных объектов через S3.

Оптимизация изменений объектов

Для оптимизации операций изменения реализовано расширение стандарта S3 в виде метода PATCH[15].

Начиная с версии 1.7.0, ПО TATLIN.OBJECT поддерживает метод PATCH, который позволяет загружать исключительно изменяемую часть объекта (дельта), что значительно оптимизирует операции модификации данных.

Принцип работы метода PATCH:

- Создается новый объект, содержащий изменения, переданные через PATCH.
- Оригинальный объект удаляется.

Метод PATCH не реализован в AWS CLI, так как является расширением стандарта S3.

Описание S3 PATCH: https://yadro.com/docs/storage/s3/api-ref/object/patch

Пример использования: драйвер CSI для K8S для интеграции с GeeseFS.

Уникальная функциональность frostfs, отсутствующая в S3

Протокол FrostFS предоставляет ряд уникальных возможностей для мониторинга и управления системой хранения:

1. **Мониторинг состояния сети**
   Получение актуальной карты сети (узлов, доступных для операций чтения/записи):

```
bash frostfs-cli netmap snapshot -r data-ip:8080 -g
```

2. **Управление контейнерами**
   Получение списка контейнерных узлов (узлов, доступных для записи в указанный бакет):

```
frostfs-cli container nodes --cid ContainerCID -r data-ip:8080 -g
```

3. **Локализация данных**
   Определение узлов хранения реплик указанного объекта:

```
frostfs-cli object nodes --oid ObjectID --cid ContainerID -r DataIP:8080 -w Wallet-file
```

4. **Работа со служебными объектами**
   Просмотр скрытых служебных объектов в контейнере (с флагом –phy):
   - Составные части большого объекта: чанки, large object, link object, MPU-чанки.
   - Tombstone: маркеры удаленных объектов (содержат OID, ключ и хеш данных)[16].
5. **Расширенный поиск объектов**
   Поиск объектов по значению атрибута (с флагом –filter). Список доступных для поиска атрибутов хранится в заголовке объекта.

   Примеры фильтров:
   - По точному совпадению (поиск MPU-чанков по Upload-ID):

```
--filter 'S3-Upload-Id EQ 12dfa52f-6b65-464b-8875-3d85a321cc82'
```

   - По точному несовпадению (исключение plain text):

```
--filter 'Content-Type NE text/plain'
```

   - По частичному совпадению (префикс S3-ключа):

```
--filter 'FilePath COMMON_PREFIX dir1/me'
```

- Отсутствие атрибута (все объекты кроме MPU-чанков):

```
--filter ' S3-Upload-Id NOPRESENT'
```

- Комбинированные условия (логическое И):

```
--filters 'S3-Upload-Id NOPRESENT' --filters 'Timestamp COMMON_PREFIX 17425'
```

6. Просмотр параметров конфигурации
   Получение системных параметров (размер чанка для больших объектов):

```
frostfs-cli netmap netinfo -r <DATA-IP>:8080 -g | grep 'object size'
Maximum object size: 67108864
```

Запуск frostfs-cli без интерактивного ввода пароля

Для автоматизации работы с утилитой frostfs-cli пароль от кошелька можно хранить в конфигурационном YAML-файле. Это позволяет избежать интерактивного ввода пароля при каждом вызове команды.

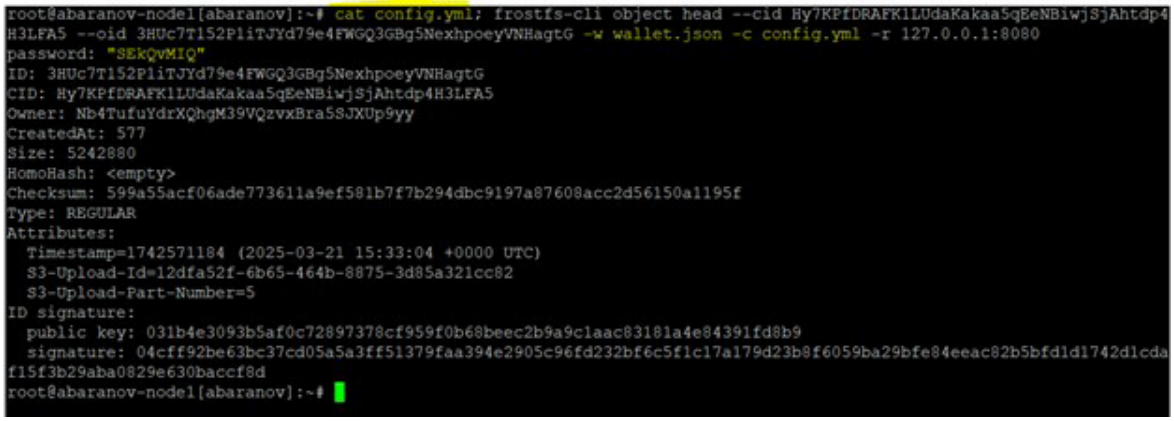
Формат конфигурационного файла

Файл config.yml содержит единственную строку с указанием пароля:

```
password: "SEkQvMIQ"
```

Пример использования

```
frostfs-cli object head \
  --cid Hy7KPfDRAFK1LUdaKakaa5qEeNBiwjSjAhtdp4H3LFA5 \
  --oid 3HUc7T152P1iTJYd79e4FWGQ3GBg5NexhpoeyVNHagtG \
  -w wallet.json \
  -c config.yml \
  -r grpc.poc.local:8080
```



Особенности создания контейнеров FrostFS в TATLIN.OBJECT 1.6

В версии TATLIN.OBJECT 1.6 произошел отказ от использования ACL (Access Control Lists) в протоколе FrostFS. Для создания контейнеров в этой версии необходимо явно устанавливать базовые ACL равными нулю:

```
frostfs-cli container create \
  --name "test-acl" \
  -r grpc.poc.local:8080 \
  -w ./wallet.json \
  --basic-acl 0x0
```

Начиная с версии TATLIN.OBJECT 1.7, опция `--basic-acl` была удалена из `frostfs-cli` и `frostfs-sdk`. Управление правами доступа к контейнерам полностью переведено на механизм APE rules (Access Policy Engine), который предоставляет более гибкую и детализированную систему управления доступом.

Ограничение доступа к данным — APE Rule

Доступ к контейнерам FrostFS и хранимым в них объектам контролируется механизмом APE Rule.

APE Rule представляет собой цепочку правил доступа, которая формируется владельцем контейнера[17]. Важно отметить, что в FrostFS правила доступа всегда применяются к конкретному контейнеру — невозможно ограничить операции пользователя в рамках всего неймспейса или системы хранения.

Приоритет применения политик:

1. Все пользователи неймспейса имеют право создавать контейнеры и управлять APE Rule для своих контейнеров.
2. Любые операции с контейнером (включая доступ для владельца) запрещены до тех пор, пока для него не задана хотя бы одна APE Rule.
3. Пользователь из другого неймспейса получает доступ к контейнеру только при наличии APE Rule, явно разрешающей доступ его UID.
4. APE Rule может предоставлять анонимный доступ к контейнеру без требования авторизации.
5. APE Rule поддерживает детализацию до уровня конкретных операций с отдельными объектами.
6. Правила применяются последовательно; в случае конфликта запрещающие правила имеют приоритет над разрешающими.
7. Цепочка правил может динамически изменяться — правила можно добавлять, удалять и модифицировать.
8. Пример установки политики public-read-write для контейнера 14E5QUfd6KK6Zn9WkyHfyyNrGXRW2UyaoE3BZ6gWmiiC:

```
frostfs-cli ape-manager add \
  --rule "allow Object.* *" \
  --rule "allow Container.Get *" \
  --target-name 14E5QUfd6KK6Zn9WkyHfyyNrGXRW2UyaoE3BZ6gWmiiC \
  --target-type container \
  -w ./wallet-ns1.json \
  --rpc-endpoint grpc.poc.local:8080
```

Особенности работы с контейнером FrostFS в именованном неймспейсе

Для создания контейнера в именованном неймспейсе требуется указание дополнительных параметров, связывающих контейнер с конкретным неймспейсом.

Пример команды создания контейнера в неймспейсе `ns3` для пользователя `iamuserns33`:

```
frostfs-cli -w ./wallet_iamuserns33.json \
  -r grpc.poc.local:8080 \
  container create \
  --name test2 \
  --nns-name iamuserns33 \
  --nns-zone ns3.ns \
  --policy "REP 2"
```

ОБЯЗАТЕЛЬНЫЕ ПАРАМЕТРЫ ДЛЯ РАБОТЫ С ИМЕНОВАННЫМИ НЕЙМСПЕЙСАМИ	
Параметр	**Описание**
–name test2	Человекочитаемое имя контейнера
–nns-name iamuserns33	Уникальная запись во внутренней системе именования (NNS). Рекомендуется использовать имя IAM-пользователя, связанного с кошельком
–nns-zone ns3.ns	Уникальная идентификация неймспейса в формате <имя_неймспейса>.ns
-r grpc.poc.local:8080	Конечная точка (endpoint) протокола FrostFS

Особенности архитектуры:

1. **Уникальность идентификаторов**
   CID контейнера уникален в пределах всей системы хранения и однозначно идентифицирует неймспейс.
2. **Единая конечная точка**
   Для работы с неймспейсами используется единая конечная точка корневого неймспейса (порт 8080).
3. **Совместимость HTTP-доступа**
   Запросы к HTTP-шлюзу с использованием CID контейнера корректно обрабатываются при обращении к HTTP-эндпоинту любого неймспейса благодаря глобальной уникальности идентификаторов контейнеров.

Таймауты при создании контейнеров/бакетов

Конфигурация бакетов/контейнеров, IAM-пользователей и настроек узлов системы хранения сохраняется в распределенной базе данных на основе блокчейна Neo. Особенностью работы данной системы является пакетная запись изменений блоками с фиксированным интервалом. Изменения вносятся в блокчейн блоками с периодичностью 8 секунд. Это означает, что операция создания нового бакета или контейнера выполняется с минимальной задержкой в 8 секунд.

При необходимости создания нескольких бакетов рекомендуется запускать их создание параллельно. Изменения, инициированные параллельно, будут объединены и записаны в блокчейн в пакетном режиме, что позволяет сократить общее время выполнения операций. Пакетная обработка изменений обеспечивает целостность и согласованность распределенной конфигурации системы, но требует фиксированного времени для подтверждения транзакций в блокчейне.

Обновление карты сети и восстановление данных отключенного узла

Узлы системы хранения формируют распределенную карту сети, которая динамически обновляется для отражения текущего состояния кластера.

Процесс исключения узла из сети:

- При потере связи с узлом система начинает мониторинг его доступности.
- Через 4-5 эпох (40-50 минут) недоступный узел исключается из актуальной карты сети.
- До момента исключения другие узлы могут пытаться проксировать запросы к отсутствующему узлу, что может временно влиять на производительность операций записи

Процесс восстановления данных:

- После исключения узла из карты сети автоматически запускается процесс репликации.
- Система восстанавливает копии объектов, которые хранились на недоступном узле, используя избыточные копии с других узлов кластера.
- Восстановление происходит в соответствии с заданной политикой хранения.

Выбор узла для доступа к контейнеру/бакету

Данные каждого бакета хранятся на заранее определенном подмножестве узлов, называемых контейнерными узлами. Система автоматически управляет маршрутизацией запросов следующим образом:

Механизм перенаправления запросов

При обращении к неконтейнерному узлу запрос автоматически перенаправляется по внутренней сети на соответствующий контейнерный узел.
Выбор конечного узла учитывает:

- принадлежность узла к целевой группе контейнерных узлов;
- доступность узла;
- локализацию данных;
- текущую загрузку ресурсов узла.

Особенности работы прокси-узла

Узел, принявший исходный запрос, действует как прокси для передачи данных.
Вносимая задержка незначительна и обычно не влияет на общую производительность.

Рекомендации по организации доступа

Для оптимального распределения нагрузки рекомендуется использовать внешний балансировщик нагрузки.
Прокси-механизм не заменяет полноценный балансировщик нагрузки, а обеспечивает базовую маршрутизацию.

Балансировка на уровне внешнего оборудования/ПО обеспечивает:

- равномерное распределение нагрузки между узлами;
- отказоустойчивость при недоступности отдельных узлов;
- оптимальное использование ресурсов кластера.

Важно: несмотря на встроенный механизм перенаправления, использование внешнего балансировщика нагрузки остается рекомендованной практикой для рабочих систем.

РАБОТА С FROSTFS-SDK

Разработка приложений с использованием протокола FrostFS осуществляется через написание кода с применением FrostFS-SDK. Исходный код SDK доступен на портале: https://git.frostfs.info/TrueCloudLab.

Особенности работы с SDK (отличия от frostfs-cli)

1. **Подключение к системе хранения**:
   Рекомендуется использовать класс pull, который реализует:
   - балансировку нагрузки на стороне клиента;
   - работу только с доступными узлами из пула подключения.
2. **Аутентификация**:
   - Для установки соединения (pull.Dial или client.Dial) может потребоваться указание WIF (формат экспорта закрытого ключа).
   - Получение адреса пользователя из файла кошелька:

```
cat <ПУТЬ_К_КОШЕЛЬКУ> | python3 -m json.tool | grep address | awk -F'"' '{print $4}'
```

   Экспорт WIF для указанного адреса:

```
neo-go wallet export -w <путь_к_кошельку> -a <адрес_пользователя>
```

   - Утилиту neo-go можно получить по ссылке: https://github.com/nspcc-dev/neo-go/releases/tag/v0.108.1
3. **Создание контейнеров**:
   - При создании контейнера параметр Basic ACL должен быть установлен в 0x00 (достаточно не указывать параметр SetBasicACL).
   - Сразу после создания контейнер недоступен для операций (включая отображение в списке контейнеров владельца).
   - Необходимо немедленно установить правила доступа с помощью AddAPEChain.
4. **Работа с контейнерами**:
   - Для систем с большим количеством контейнеров рекомендуется использовать ListContainersStream вместо ListContainers.
   - Получение полного списка объектов осуществляется через операцию поиска SearchObjects (отдельной операции полного листинга не предусмотрено).
5. **Модификация объектов**:
   - Изменение и дополнение информации в объекте выполняется с помощью метода Patch

Примеры реализации:

Пример для frostfs-sdk-go (TATLIN.OBJECT 1.7): https://git.frostfs.info/KurlesHS/frostfs-sdk-go-example
Пример для frostfs-sdk-java (TATLIN.OBJECT 1.7): https://git.frostfs.info/orikik/frostfs-sdk-java-example

ОСОБЕННОСТИ РАБОТЫ С S3

Управление доступом через IAM Policy

Доступ к бакетам и объектам в системе TATLIN.OBJECT обеспечивается механизмом IAM Policy (Identity and Access Management).

Уровни применения политик:

- Уровень пользователя (IAM User Policy) — политики, применяемые к конкретному пользователю.
- Уровень бакета (Bucket Policy) — политики, применяемые к конкретному бакету.

Приоритет применения политик доступа:

1. Права на административные операции (управление пользователями неймспейса, создание бакетов) могут быть предоставлены только через IAM User Policy.
2. IAM User Policy применяется только к бакетам в рамках того неймспейса, к которому принадлежит пользователь.
3. В пределах одного неймспейса действуют следующие правила приоритета:
   - если IAM User Policy разрешает операцию, а Bucket Policy запрещает — операция разрешается.
   - если IAM User Policy запрещает операцию, а Bucket Policy разрешает — операция запрещается.
   - если IAM User Policy не содержит правил для операции, права определяются Bucket Policy.
4. При отсутствии Bucket Policy права на операции с бакетом имеют:
   - Пользователь, создавший бакет.
   - Пользователи, имеющие соответствующие разрешения в своих IAM User Policy.
5. Bucket Policy может предоставлять анонимный доступ к бакету без требования авторизации.

Оптимизация удаления данных в S3

В TATLIN.OBJECT версии 1.7 реализованы несколько механизмов для эффективного удаления данных через протокол S3:

1. Lifecycle Policy (Политика жизненного цикла)
   - Стандартный инструмент S3 для автоматизированного удаления объектов и их версий по временным меткам.
   - Позволяет задавать различные сроки хранения для разных префиксов (каталогов) в рамках одного бакета.
   - Обеспечивает плановое автоматическое удаление данных без ручного вмешательства.
2. Быстрое удаление бакета через S3 REST API[18]
   - Для немедленного удаления бакета со всем содержимым необходимо добавить в заголовок запроса специальное поле:

     ```
     X-Amz-Force-Delete: true
     ```

   - Принцип работы: на низком уровне происходит удаление FrostFS-контейнера, соответствующего бакету, без поэтапного удаления отдельных объектов.
   - Особенности реализации:
     - Бакет удаляется моментально.
     - Фактическое удаление объектов данных выполняется сборщиком мусора в фоновом режиме[19].
     - Значительно сокращается время выполнения операции по сравнению с последовательным удалением объектов.

ETag S3-объекта

ETag — это атрибут объекта, содержащий его криптографический хеш. Значение ETag совпадает у идентичных объектов и используется для:

- Сравнения объектов в системе хранения (включая разные системы).
- Проверки соответствия загружаемого объекта существующему на системе.
- Контроля целостности скачанных данных.

Алгоритм расчета ETag:

- Для стандартной загрузки: вычисляется MD5 или SHA-256 хеш содержимого объекта.
- При использовании шифрования: хеш рассчитывается от зашифрованных данных.

Особенности расчета для многочастной загрузки (MPU)[20]:

- Вычисляется ETag для каждой отдельной части (MPU-чанка).
- Полученные значения хешей объединяются в единую строку.
- Вычисляется ETag от результирующей строки.
- К полученному значению через дефис добавляется общее количество частей.

Сценарии, в которых расчет ETag невозможен или некорректен:

- Использование серверного шифрования (Server-Side Encryption), начиная с версии 1.7.2.
- Применение операции Patch для модификации объекта.
- Использование алгоритмов, отличных от MD5 или SHA-256 (спецификация Amazon допускает различные алгоритмы).

Ссылка на спецификацию Amazon:
https://docs.aws.amazon.com/AmazonS3/latest/API/API_CompleteMultipartUpload.html

Листинг бакета с большим числом объектов

В версии 1.6 системы при выполнении листинга содержимого бакета с разных узлов могли возвращаться неполные или отличающиеся списки объектов. Это было связано с задержкой синхронизации базы данных сервисного слоя между контейнерными узлами. Недавно измененные, удаленные или созданные объекты могли отображаться только на части узлов.
Для обеспечения консистентности листинга рекомендовалось ограничить операции записи и чтения одним узлом системы с помощью настройки session-persistence на балансировщике нагрузки.

В версии 1.7 системы архитектура была улучшена:

- Для каждого бакета автоматически назначается главный контейнерный узел, который первым получает в се обновления базы данных сервисного слоя.
- Остальные контейнерные узлы синхронизируют данные с главного узла.
- Запросы на листинг автоматически направляются на главный узел, что обеспечивает консистентность данных.

Рекомендации по работе с большими бакетами:

1. **Использование API**

```
aws s3api list-objects-v2 --bucket bucket-name
```

2. **Применение пагинации**

```
aws s3api list-objects-v2 --bucket bucket-name --page-size 5000
```

Рекомендуемый размер страницы: 1000-10000 объектов.
Подбор оптимального значения осуществляется экспериментально.

3. **Настройка таймаутов**

```
aws s3api list-objects-v2 --bucket bucket-name \
  --cli-connect-timeout 60 \
  --cli-read-timeout 120
```

Увеличение времени ожидания для обработки больших объемов данных.

Совместимость со сторонними S3 клиентами

Сторонние решения, совместимые с S3, могут иметь различные требования к реализации протокола Amazon S3. Для обеспечения совместимости с TATLIN.OBJECT рекомендуется:

- Использовать path-style адресацию (например, https://s3.example.com/bucket-name/key).
- Применять подпись формата AWS SigV4[21] или AWS SigV4E.
- Использовать для контроля доступа Bucket Policy вместо объектных ACL.
- Подключаться через порт 443 с обязательным использованием TLS.

Диагностика проблем совместимости

При возникновении проблем с конкретным клиентским ПО рекомендуется проверить:

1. **Поддержку методов S3**
   Соответствие методов, указанных в Руководстве разработчика TATLIN.OBJECT.
2. **Требования к форматам заголовков**
   Наличие специфических полей заголовков[22].
   Алгоритмы вычисления подписей и контрольных сумм.
3. **Обработку служебной информации**
   Некоторые клиенты (например, boto3 1.36.0) добавляют служебную информацию в начало / конец объектов. Это может вызывать проблемы при чтении объектов другими клиентами.
4. **Учет особенностей хранения**
   S3-to-FS решения должны учитывать иммутабельность объектов.
   Рекомендуется использование метода PATCH для модификации данных.
5. **Поддержку расширенных функций**
   Частичная поддержка WORM (S3 Object Lock).
   Обработка объектов, превышающих лимиты AWS S3.
   Толерантность к eventually consistent листингу.
6. **Ограничения клиентского ПО**
   Поддерживаемая длина S3-ключа.
   Требования к формату учетных данных (может потребоваться импорт через CAPI).
7. **Настройки таймаутов**
   Увеличение таймаутов при операциях листинга больших бакетов.
   Настройка повторных попыток соединения.

Неподдерживаемые функции:

- Тиринг бакетов (bucket tiering).
- Смена региона хранения.
- Полная поддержка S3 Object Lock.

Использование curl для работы с S3

Для работы с TATLIN.OBJECT через S3 API curl должен быть версии 8.0 или выше (обязательная поддержка флага –aws-sigv4).
Формат команды:

```
curl --head \
  -s \
  -k \
  --user <accesskey>:<secretkey> \
  --aws-sigv4 "aws:amz:<region>:s3" \
  https://${s3-endpoint}/${bucket}/${file}
```

Обязательные параметры:

- `--user <accesskey>:<secretkey>` — учетные данные S3 (ключ и секрет).
- `--aws-sigv4 "aws:amz:<region>:s3"` — использование подписи AWS Signature Version 4.
- `<region>` — location constraint (например, `default`)[23].
- `https://${s3-endpoint}:443/${bucket}/${file}` — полный URL целевого объекта.

Опциональные параметры:

- `--head` — выполнение HEAD-запроса (получение метаданных объекта).
- `-s` — тихий режим (отключение вывода служебной информации и индикатора прогресса).
- `-k` — игнорирование проверки SSL-сертификата (не рекомендуется для использования в рабочих средах).

Пример команды:

```
curl --head \
  -s \
  -k \
  --user AKIAIOSFODNN7EXAMPLE:wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY \
  --aws-sigv4 "aws:amz:default:s3" \
  https://s3-storage.example.com/mybucket/document.pdf
```

УСКОРЕНИЕ ОДНОПОТОЧНЫХ ЗАГРУЗОК ФАЙЛОВ ПО ССЫЛКАМ PRESIGNED-URL

Существует возможность ускорения однопоточных загрузок файлов на конечных клиентских устройствах. Например, браузер Chrome по умолчанию не может целиком задействовать весь канал для скачивания файлов для presigned-url ссылок, оставляя запас скорости для остальных программ. Но он имеет скрытую настройку «Parallel downloading» («Параллельная загрузка»), позволяющую снять это ограничение и использовать для подключения к S3-серверу многопоточную параллельную загрузку.

Настройка находится в скрытом меню Flags. Для доступа к ней, введите в адресной строке:

```
chrome://flags
```

или

```
chrome://flags/#enable-parallel-downloading
```

После открытия меню найдите опцию «Parallel downloading» и измените значение в поле напротив нее с «Default» («По умолчанию») на «Enabled» («Включено»). После этого в нижней части экрана появится всплывающее уведомление о том, что изменения вступят в силу после перезапуска браузера. Нажмите кнопку «Relaunch» («Перезапустить»), чтобы перезапустить браузер и сохранить настройки. Теперь браузер будет использовать многопоточную загрузку при скачивании файлов, разбивая их на несколько частей, и загрузка будет происходить значительно быстрее.

ОБЩИЕ РЕКОМЕНДАЦИИ ДЛЯ ОПТИМИЗАЦИИ ОПЕРАЦИЙ ЗАПИСИ/ЧТЕНИЯ

1. **Настройка сети**:
   На этапе планирования и развертывания системы хранения выберите максимально возможное значение MTU, поддерживаемое вашей сетевой инфраструктурой.
   Использование MTU 9000 обеспечивает до 10 % прироста производительности по сравнению со стандартным MTU 1500.
2. **Георазпределенные конфигурации**:
   Используйте асинхронные политики хранения с созданием синхронных копий только в пределах локальной площадки.
   Важно: чем больше синхронных копий требуется создать, тем ниже производительность операций записи. Для операций записи и чтения используйте узлы локальной геоплощадки (с наименьшей сетевой задержкой до клиента).
3. **Мониторинг и обслуживание**:
   Регулярно отслеживайте состояние системы хранения, обращая особое внимание на появление неисправностей и аномалий (например, переход шардов в состояния, отличные от READ-WRITE).
   Все отклонения от нормальной работы требуют своевременного обращения в сервисную службу YADRO.
4. **Оптимизация операций модификации**:
   Для обновления содержимого объектов используйте метод PATCH, который позволяет загружать только изменяемую часть (дельту), а не весь объект целиком. Это значительно снижает нагрузку на сеть и систему хранения.

[1] Руководство по многочастной загрузке в AWS S3:
https://repost.aws/knowledge-center/s3-multipart-upload-cli

[2] Спецификация загрузки частей объектов:
- RFC 9110, раздел Range: https://www.rfc-editor.org/rfc/rfc9110.html#name-range,
- Документация AWS: https://docs.aws.amazon.com/AmazonS3/latest/userguide/download-objects.html

[3] Функциональность доступна только при наличии SSD-ресурсов в конфигурации шардов узла.

[4] В версии 1.7 диапазон размеров объектов, обрабатываемых через кэш, изменен с 8 КиБ–2 МиБ на 128 КиБ–2 МиБ.

[5] Для работы требуется наличие SSD-ресурсов в конфигурации шардов узла.

[6] В версии 1.7 лимит размера объекта для кэширования изменен с 8 МиБ на 2 МиБ.

[7] Чанки одного объекта могут распределяться для хранения на разных шардах.

[8] Ограничение на размер объекта в 5 ГиБ определено спецификацией Amazon S3:
https://aws.amazon.com/s3/faqs/#:~:text=Individual%20Amazon%20S3%20objects%20can,using%20the%20multipart%20upload%20capability

[9] В версии 1.6.2 загрузка прерывается по таймауту через 100 минут.

[10] В FrostFS подготовка чанков выполняется на стороне клиента с использованием SDK. Загрузка подготовленных чанков осуществляется клиентским ПО.

[11] Протокол S3 ограничивает количество частей при многочастной загрузки значением 10 000:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/qfacts.html.

[12] К загружаемым через MPU объектам применяются те же правила, что и к обычным объектам S3. Например, MPU объект размером 65 МиБ будет разделен на чанки 64 МиБ и 1 МиБ, а также будут созданы служебные объекты large object и link object.

[13] Ограничение в 10 000 частей на одну MPU-загрузку:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/qfacts.html.

[14] В версиях 1.6/1.7 изменения, выполненные через FrostFS API, не синхронизируются с сервисным слоем и не отображаются на уровне S3. Полная мультипротокольная поддержка запланирована в будущих версиях.

[15] Особенности реализации PATCH:
- Yandex S3 запрещает использование PATCH в версионированных бакетах.
- TATLIN.OBJECT не имеет этого ограничения.
- FrostFS API метод PATCH сохраняет как оригинальный, так и измененный объект.

[16] Маркеры удаления (tombstone) хранятся на шардах семь дней (в некоторых случаях дольше). Наличие tombstone предотвращает создание объекта с идентичными OID, содержимым и заголовками.

[17] Документация APE Rule:
https://git.frostfs.info/TrueCloudLab/policy-engine/src/branch/master/docs/ape.md

[18] Спецификация API DeleteBucket:
https://docs.aws.amazon.com/AmazonS3/latest/API/API_DeleteBucket.html

[19] Процесс удаления объектов:
- Для каждого удаляемого объекта создается маркер tombstone.
- Сборщик мусора в фоновом режиме обрабатывает tombstone и удаляет связанные объекты.
- Tombstone старше семи дней удаляются сборщиком мусора.
- R&D может восстановить объект до его физического удаления.

[20] Алгоритм расчета ETag для многочастных загрузок: ETag не является MD5-хешем всего объекта, а вычисляется из конкатенированных MD5-хешей отдельных частей с добавлением количества частей.

[21] Поддержка алгоритмов подписи:
- s3curl не поддерживается (только SigV2).
- AWS CLI 1.* требует принудительного включения SigV4.

[22] Проблемы совместимости:
- AWS CLI 2.23 использует алгоритм crc64nvme, несовместимый с большинством S3-сервисов.
- Требуется отключать проверку контрольных сумм.

[23] Руководство по подписанию запросов AWS SigV4 с помощью curl:
https://how.wtf/aws-sigv4-requests-with-curl.html