РЕСПУБЛИКА КАЗАХСТАН

(19) **KZ** (13) **B** (11) **38142**
(51) ***H04L 67/00*** (2006.01)

МИНИСТЕРСТВО ЮСТИЦИИ РЕСПУБЛИКИ КАЗАХСТАН

ОПИСАНИЕ ИЗОБРЕТЕНИЯ
К ПАТЕНТУ

(21) 2026/0326.1
(22) 14.04.2026
(45) 10.07.2026, бюл. №27
(72) Ткаченко Артур Григорьевич
(73) Товарищество с ограниченной ответственностью «Kaspi Магазин»
(56) US 9723065 B2, 01.08.2017
US 12314134 B2, 27.05.2025
US 20190370362 A1, 05.12.2019

(54) **СИСТЕМА И СПОСОБ ИНТЕГРАЦИИ ПРИКЛАДНЫХ ПРОГРАММНЫХ СИСТЕМ-ПОТРЕБИТЕЛЕЙ И ИНФРАСТРУКТУРЫ ХРАНЕНИЯ ДАННЫХ**

(57) Система и способ интеграции прикладных программных систем-потребителей и инфраструктуры хранения данных относятся к области информационно-коммуникационных технологий. Сущность изобретения заключается в следующих существенных признаках: центральная компьютерная станция с прокси-сервером, проверяющим соответствие загружаемых файлов заявленным типам данных и поддерживающим работу с учетными записями; виртуализация; динамические алиасы; встроенный фильтр макросов; гибридная инфраструктура; кастомный Python в runtime; отказоустойчивая Master-Master репликация со Snapshot-восстановлением; Deep Copy и защита целостности Rich Text ассетов; виртуальный S3-агрегатор с динамическим маппингом полномочий; системная интеграция, использование инфраструктуры Copy-on-Write.

Задача изобретения – создание наиболее функционального интеллектуального шлюза между прикладными системами-потребителями и физической инфраструктурой хранения.

Технический результат – повышение уровня безопасности, динамической трансформации и доставки контента S3-гейтвея.

(19) KZ (13) B (11) 38142

Изобретение относится к области информационно-коммуникационных технологий, а именно – области управления медиа-активами и данными путем интеграции прикладных систем-потребителей и физической инфраструктуры хранения на основе S3-гейтвея (шлюза, прокси-сервера). Иными словами, система и способ применяются для организации единого узла обработки и маршрутизации медиа-файлов и офисных документов между различными прикладными системами и аппаратными серверными мощностями.

Все прикладные системы-потребители взаимодействуют с заявленной системой через стандартный протокол S3. Система позволяет использовать физически разрозненные S3-хранилища (включая разные центры обработки данных и/или провайдеров). Поддержка виртуальных S3-реквизитов позволяет гибко распределять права доступа на уровне гейтвея, даже если целевое хранилище имеет ограничения по количеству ключей или политик. Балансировка запросов и выбор целевого хранилища настраиваются через административную панель без внесения изменений в программный код систем-потребителей. Имеется встроенный браузер объектов с функциями поиска и предпросмотра.

Система берет на себя базовый контур защиты при приеме файлов, а именно: автоматически выявляет и блокирует специфические угрозы (ZIP/JPEG-бомбы); гарантирует соответствие загружаемых файлов заявленным типам данных; поддерживает работу с клиентским и партнерским SSO, а также доменными учетными записями сотрудников – для предоставления доступа к контенту отдельным клиентам, партнерам или группам сотрудников.

Система включает в себя масштабируемый вычислительный движок обработки (Processing Engine):

1) Real-time – для задач, требующих мгновенного результата (например, генерация превью);

2) Async (Kafka) – для фоновых задач, не требующих мгновенного ответа;

3) Jobs – для тяжелых вычислительных операций, включая, но не ограничиваясь, работой с Large Language Model (LLM).

Форматы и методы обработки, включая, но не ограничиваясь, масштабированием, обрезкой и изменением разрешения, привязываются к бакетам через административную панель. Поддерживается внедрение кастомной логики на Python без перезапуска сервисов. Система характеризуется динамической генерацией: формирование изображений на основе шаблонов (JSON + SVG), генерация Rich Content (HTML из формата JSON), наложение водяных знаков.

В системе используется Valkey для блочного кэширования медиа-контента в оперативной памяти (Range-requests), что оптимизирует нагрузку на каналы связи при работе с видео.

Надежность и отказоустойчивость изобретения обеспечиваются следующими функциями:

1) в случае недоступности объекта в основном хранилище система автоматически пытается восстановить его из доступных резервных аппаратных источников;

2) при временной недоступности целевых хранилищ данные сохраняются и реплицируются позже, гарантируя сохранность информации.

Существуют частичные аналоги системы – облачные прокси, специализированные хранилища. Между тем заявленное изобретение уникально за счет объединения в одном контуре функций безопасности, динамической трансформации и доставки контента, а также механизмов аппаратной оптимизации ресурсов.

Наиболее близким к изобретению аналогом является патент US 12,132,807 B2 на «THIRD PARTY GATEWAY». Представляет собой шлюз «третьей стороны» для проверки исходящего трафика в компьютерной сетевой системе, причем шлюз «третьей стороны» содержит: прокси-сервер; агент шлюза. Прокси-сервер настроен для терминирования TLS-соединения (Transport Layer Security) для исходящего трафика, дополнительно настроен для пересылки исходящего трафика агенту шлюза. Агент шлюза настроен для проверки исходящего трафика как допустимого или недопустимого на основе заранее определенных требований проверки. Прокси-сервер дополнительно настроен для получения допустимого или недопустимого трафика от агента шлюза и для пересылки допустимого трафика целевой службе или для блокирования недопустимого трафика. Агент шлюза настроен для выполнения проверки учетной записи для исходящего трафика. Между тем рассматриваемый аналог – это статичный инспектор, который просто «пропускает или блокирует» пакет, в то время как заявленное изобретение – это динамический контроллер, который создает виртуальную инфраструктуру поверх физических хранилищ. В этой связи аналог не обладает следующими функциями системы и способа: виртуализация; использование алиасов; обеспечение высокого уровня безопасности за счет встроенного фильтра макросов; обеспечение гибридной инфраструктуры (Self-hosted + Cloud); обеспечение высокого уровня гибкости логики за счет кастомного Python в runtime.

Для виртуализации используется архитектура Copy-on-Write (CoW) для тестов и дельт – реализован механизм доступа к актуальным данным из продуктового контура без их физического копирования. На дисковом пространстве тестовой среды создается только «дельта» (измененные данные), что существенно экономит аппаратные ресурсы.

Использование динамических алиасов (SKU, ID заказа) обеспечивает S3-гейтвею функционирование в режиме реального времени. За счет подмены алиасов «на лету» (а также за счет динамической балансировки запросов) полностью скрывается реальная структура сети от клиента. Таким образом, система является агрегатором, объединяющим физически разрозненные хранилища (разные центры обработки данных и провайдеры) в единую точку

входа, в то время как аналог является лишь фильтром для одного канала связи и не умеет управлять распределенной инфраструктурой.

Встроенный фильтр макросов представляет собой проверку входящих офисных документов на наличие исполняемых скриптов. При обнаружении потенциально опасного содержимого объект блокируется и маркируется соответствующим образом для последующей обработки специализированным программным обеспечением.

Гибридная инфраструктура позволяет сократить затраты на содержание аппаратных мощностей, эффективно хранить разные типы данных и легко масштабироваться.

Гибкость логики за счет кастомного Python в runtime проявляется в возможности динамически изменять поведение вычислительных узлов системы без их перезапуска. Это позволяет динамически менять логику обработки и валидации объектов, в то время как аналог ограничен набором статичных правил проверки.

Помимо вышеуказанного, изобретательский уровень системы и способа обеспечивается следующими функциями: отказоустойчивая Master-Master репликация со Snapshot-восстановлением; Deep Copy и защита целостности Rich Text ассетов; виртуальный S3-агрегатор с динамическим маппингом полномочий; системная интеграция (Синергия).

Отказоустойчивая Master-Master репликация со Snapshot-восстановлением представляет собой систему двусторонней синхронизации данных между центрами обработки данных и механизмом самовосстановления через Snapshot-транзакции. В стандартных системах при обрыве связи очередь репликации заполняется ошибками, и при восстановлении система пытается повторить все устаревшие действия по порядку, что ведет к лавинообразной нагрузке на процессоры и риску перезаписи свежих данных старыми. Для устранения этого недостатка система фиксирует объект как «проблемный» и приостанавливает генерацию ошибок в очереди (Skip Logic), а после восстановления связи запрашивает текущий «слепок» данных (Snapshot), игнорируя промежуточные состояния. Это не просто «повтор запроса» (retry), а интеллектуальное приведение системы к финальному согласованному состоянию (Eventual Consistency) с минимизацией сетевого трафика.

Deep Copy и защита целостности Rich Text ассетов – это автоматическое клонирование вложенных медиа-файлов при копировании JSON-документов. В обычных S3-хранилищах файл – это просто набор байтов. При копировании текстового файла (JSON), содержащего ссылки на изображения, копируется только текст. Если оригинал изображений удалят, копия документа станет недействительной. В изобретении процессор шлюза анализирует структуру файла (JSON) в момент копирования, извлекает список зависимостей и физически переносит их в новый бакет, обеспечивая автономность копии на дисковом накопителе. Таким образом, система и способ обеспечивают анализ контента, а не трафика. Если вышеуказанный аналог инспектирует сетевые пакеты, заявленная система осуществляет глубокий парсинг структуры файлов.

Виртуальный S3-агрегатор с динамическим маппингом полномочий представляет собой единую аппаратную точку входа для множества физических S3-кластеров с подменой ключей доступа. Процессор шлюза принимает запросы по «виртуальным» ключам (виртуальным S3-реквизитам доступа), самостоятельно определяет, в каком физическом хранилище лежит бакет, и динамически переподписывает запрос реальными ключами этого хранилища, подменяя заголовки и пути.

Под системной интеграцией (Синергией) понимается интеграция всех вычислительных слоев – Storage, Processing, Metadata, Replication. Согласно устоявшейся практике, для репликации используется одно программное обеспечение, для обработки медиа – другое (например, OpenCV), для Rich Text – третье, а для S3-проксирования – четвертое. Это ведет к потере контекста: S3-прокси не знает, что внутри JSON-файла лежат ссылки на изображения, а система репликации не умеет восстанавливать поврежденные метаданные при сетевых ошибках. Указанный недостаток устраняется за счет следующих примеров функционирования изобретения:

1) поскольку шлюз взаимодействует с сервисом метаданных, он может выполнять валидацию документов непосредственно в момент загрузки в оперативную память;

2) благодаря тому, что логика репликации интегрирована с бизнес-логикой объектов, вычислительные узлы могут использовать Snapshot-транзакции вместо простого повтора HTTP-пакетов.

Таким образом, сущность заявленного изобретения заключается в совокупности следующих существенных признаков: центральная компьютерная станция с прокси-сервером, проверяющим соответствие загружаемых файлов заявленным типам данных и поддерживающим работу с учетными записями; виртуализация; динамические алиасы; встроенный фильтр макросов; гибридная инфраструктура (Self-hosted + Cloud); кастомный Python в runtime; отказоустойчивая Master-Master репликация со Snapshot восстановлением; Deep Copy и защита целостности Rich Text ассетов; виртуальный S3-агрегатор с динамическим маппингом полномочий; системная интеграция (Синергия), использование инфраструктуры Copy-on-Write (CoW).

Задачей изобретения является создание наиболее функционального интеллектуального шлюза между прикладными системами-потребителями и физической инфраструктурой хранения.

Технический результат заключается в повышении уровня безопасности, динамической трансформации и доставки контента S3-гейтвея.

Возможность осуществления изобретения подтверждается нижеследующей последовательностью работы подсистем.

1. Виртуализация учетных записей и авторизация. Система обеспечивает абстракцию прикладного слоя от физических параметров безопасности хранилищ через механизм виртуализации реквизитов доступа. Это позволяет централизованно управлять политиками безопасности и обходить технические лимиты различных облачных провайдеров (фигура 1).

Шаг 1.1: Прием запроса через виртуальные реквизиты. Прикладная система направляет стандартный S3-запрос к гейтвею, используя виртуальные S3-ключи (Access/Secret keys). Эти ключи являются логическими сущностями внутри системы и не имеют прямого соответствия ключам в конечных физических хранилищах данных.

Шаг 1.2: Динамический маппинг полномочий. Процессор гейтвея выполняет сопоставление виртуального ключа с реальными учетными данными целевого физического хранилища (бакета). Система определяет права доступа на уровне шлюза, что позволяет гибко распределять полномочия, даже если целевое хранилище имеет ограничения по количеству ключей или сложности политик доступа.

Шаг 1.3: Трансформация и переподпись запроса. Гейтвей динамически выполняет переподпись заголовков авторизации в соответствии с протоколом Signature V4 (или иным требуемым протоколом целевого провайдера). В процессе переподписи происходит подмена оригинальных заголовков аутентификации на валидные реквизиты конкретного центра обработки данных или облачного сервиса.

Шаг 1.4: Прозрачное управление ключами. Данный механизм позволяет прикладным системам работать с неизменным («единым») набором ключей, в то время как администратор гейтвея может ротировать или изменять ключи на Backend без уведомления систем-потребителей и внесения правок в их программный код.

2. Виртуальный S3-агрегатор и подмена путей. После прохождения этапа авторизации система определяет физическое местоположение объекта, полностью абстрагируя его от логической структуры запроса прикладной системы. Этот механизм позволяет реализовать концепцию «невидимой инфраструктуры», где приложение взаимодействует с единым интерфейсом, не обладая информацией о реальной топологии сети (фигура 2).

Шаг 2.1: Трансформация логической структуры. Система выполняет прозрачную подмену целевых бакетов и путей. Данный логический путь выполняет функцию динамического алиаса. Например, путь в запросе /user-data/avatar.jpg может быть перенаправлен гейтвеем в глубокую иерархию физического дискового хранилища, такую как dc-almatystorage/legacy-projects/v1/avatars/, без уведомления клиента об изменении структуры.

Шаг 2.2: Автоматический выбор целевого хранилища. В случае наличия в целевом контуре (центре обработки данных) нескольких разрозненных S3-инсталляций, вычислительный узел гейтвея автоматически выбирает конечное хранилище на основе заданных политик и текущих настроек балансировки. Это обеспечивает гибкое распределение данных между различными серверными мощностями и провайдерами.

Шаг 2.3: Полная мутация URL-адреса. Гейтвей осуществляет динамическую модификацию конечной точки запроса (endpoint), обеспечивая корректную маршрутизацию в сложной гибридной инфраструктуре, сочетающей Self-hosted и Cloud решения.

Шаг 2.4: Изоляция конфигурации. Весь процесс выбора хранилища и изменения пути происходит на уровне базы данных административной панели гейтвея, что исключает необходимость внесения изменений в конфигурационные файлы систем-потребителей.

3. Интеллектуальный прием и многоуровневая валидация (Ingress). Система реализует глубокую инспекцию входящего потока данных в оперативной памяти для обеспечения безопасности инфраструктуры хранения. Решение предотвращает сохранение вредоносного контента на жесткие диски на этапе загрузки (фигура 3).

Шаг 3.1: MIME-валидация и структурный анализ. Система выполняет проверку файлов на основе анализа магических байтов (magic bytes). На этом этапе также происходит выявление и блокировка специфических угроз (включая, но не ограничиваясь, ZIP- и JPEG-бомбами), способных вызвать критическую нагрузку на процессор (отказ в обслуживании). Шаг 3.2: Интеграция с антивирусным программным обеспечением. Реализовано взаимодействие с системами защиты через протоколы ICAP и KESL (Kaspersky Endpoint Security for Linux) REST API для сканирования объектов.

Шаг 3.3: Защита по таймауту и предотвращение DOS. Действует строгий лимит процессорного времени на сканирование. Если операция превышает заданный порог, процессор принудительно прерывает ее, а объект признается небезопасным.

Шаг 3.4: Глубокий фильтр макросов. Встроенный анализатор проверяет документы, включая, но не ограничиваясь, форматами Microsoft Office, OpenOffice и PDF, на наличие активного содержимого (VBA-скриптов и «подозрительных» макросов).

Шаг 3.5: Режимы изоляции трафика. Система поддерживает два сценария:

1) In-memory обработка: Проверки проводятся исключительно в оперативной памяти, без записи на диск. Если объект опасен, сессия загрузки обрывается.

2) Логическая блокировка: Объект сохраняется на накопитель, но маркируется и блокируется на уровне гейтвея, до подтверждения безопасности администратором.

4. Динамическая трансформация и Processing Engine. Масштабируемый вычислительный движок обработки данных (Processing Engine) позволяет управлять медиа-контентом без изменения программного кода прикладных систем. Процесс постановки задач и генерации контента отражен на фигуре 4, а алгоритм последующего поиска и

адаптивной выдачи сформированных объектов – на фигуре 5.

Шаг 4.1: Стратегии обработки. Вычислительные узлы системы поддерживают четыре метода выполнения задач трансформации: синхронно на шлюзе (в оперативной памяти в процессе загрузки); асинхронно через Kafka (передача фоновым процессам); асинхронные задания (Jobs) для ресурсоемких операций; в режиме по запросу (On-demand), когда артефакты генерируются при обращении пользователя без обязательного сохранения на диск.

Шаг 4.2: Управление и версионность настроек. Параметры обработки привязываются к бакетам через административную панель и применяются процессорами узлов немедленно. Шаг 4.3: Снапшот настроек объекта. Для уже обработанных объектов система фиксирует в метаданных «слепок» настроек, актуальных на момент их создания. При переобработке система анализирует старый снапшот и переиспользует существующие на диске артефакты для экономии вычислительных ресурсов.

Шаг 4.4: Стратегия миграции на больших данных. При изменении политик система запускает фоновое задание (Job), которое аппаратно пересчитывает исторические данные в новый вид, при этом старые объекты остаются доступными по исходным путям до окончания миграции.

Шаг 4.5: Гарантия наличия артефактов. Режим Real-time обеспечивает создание производных объектов непосредственно до отправки клиенту подтверждения об успешной транзакции.

Шаг 4.6: Динамическое исполнение пользовательской логики (кастомный Python в runtime). Исполняемые скрипты, заданные через интерфейс администратора, сохраняются в базе данных системы. При сохранении или обновлении скрипта процессоры всех вычислительных узлов динамически перечитывают обновленный код на языке Python и инициализируют соответствующие классы на его основе. Выполнение указанных скриптов осуществляется в изолированной среде GraalPy (на базе GraalVM 25) с использованием технологии Polyglot. Это позволяет применять произвольную бизнес-логику к обрабатываемым объектам в точках, указанных в настройках, без перезапуска основных служб. В качестве примера практического осуществления, включая, но не ограничиваясь указанным сценарием, при обработке изображений скрипт растягивает краевые пиксели и накладывает поверх прозрачный градиент для плавного перехода в фоновый цвет. Для обеспечения высокой пропускной способности (Ingress) и оптимизации передачи данных между основной вычислительной средой (Java) и изолированной средой скриптов (GraalPy) реализована прямая работа с массивами байтов (byte[]). Это позволяет процессору обрабатывать потоки данных напрямую, избегая излишнего копирования в памяти и копирования ресурсоемких промежуточных преобразований (таких как конвертация в списки формата ForeignList).

Шаг 4.7: Оптимизация видеоконтента. Используется высокопроизводительное блочное кэширование фрагментов видеофайлов в оперативной памяти (Valkey), снижающее нагрузку на сетевые интерфейсы.

5. Deep Copy и защита целостности Rich Content. Для защиты сложного контента, содержащего текстовую информацию с внедренными медиа-активами, в системе применяется механизм глубокого копирования на уровне физических носителей (фигура 6). Шаг 5.1: Анализ структуры Rich Content. При копировании процессор гейтвея анализирует структуру документа (например, JSON), извлекая список зависимостей, включая, но не ограничиваясь, изображениями и видеофайлами, на которые ссылается текст.

Шаг 5.2: Автономное клонирование зависимостей. Система выполняет физическое копирование байтов всех выявленных медиа-активов в целевой бакет на дисковом массиве, гарантируя, что копия останется рабочей при удалении оригиналов.

Шаг 5.3: Адаптивное применение форматов при генерации артефактов. Система автоматически применяет политики обработки целевого бакета: генерирует базовые медиа для черновых бакетов или запускает генерацию полного набора адаптивных разрешений (DPI) при переносе в публичный бакет.

Шаг 5.4: Защита целостности ассетов. Интеграция слоев хранения и вычислений предотвращает появление недействительных ссылок в скопированных документах.

Шаг 5.5: Применение настроек форматов цели.

Шаг 5.6: Финализация и подмена ссылок в JSON.

6. Отказоустойчивая репликация и разрешение конфликтов. Синхронизация данных в распределенной среде (Master-Master) исключает лавинообразную сетевую нагрузку после аппаратных сбоев (фигура 7).

Шаг 6.1: Разрешение конфликтов через DVV (Dot-Version Vectors). Вычислительные узлы используют векторы версий для определения актуальной ревизии данных при одновременной записи в разных дата-центрах.

Шаг 6.2: Применение механизма Skip Logic. При обрыве связи система маркирует пакет данных как «проблемный» и приостанавливает циклы повторных запросов к нему процессором, предотвращая блокировку очереди.

Шаг 6.3: Snapshot-восстановление. После восстановления связи процессор запрашивает актуальный «слепок» состояния дисковой подсистемы (Snapshot), игнорируя промежуточные транзакции.

Шаг 6.4: Трансформирующая репликация. Для экономии пропускной способности каналов связи между центром обработки данных могут реплицироваться не оригинальные файлы, а их исключительно оптимизированные аппаратным движком версии.

7. Внутренняя репликация и балансировка нагрузки. В рамках одного центра обработки

данных реализован контур оптимизации дисковой подсистемы (фигура 8).

Шаг 7.1: Гибкая Master-Slave архитектура. Реализована селективная репликация: на ведомые серверы передаются не все данные, а их конкретные наборы в зависимости от частоты обращений.

Шаг 7.2: Оптимизация хранения через форматы. «Горячие» артефакты (превью) кэшируются на высокоскоростных NVMe/SSD накопителях ведомых узлов. Тяжелые оригиналы маршрутизируются на узлы с емкими, но менее скоростными HDD-накопителями («холодное» хранение).

Шаг 7.3: Балансировка по чтению (Read-only Balancing). Процессор маршрутизирует запрос к тому физическому серверу, на дисках которого уже сформирован нужный артефакт. Шаг 7.4: Обеспечение высокой доступности. При выходе из строя сервера-реплики гейтвей перенаправляет трафик или инициирует генерацию необходимого производного файла динамически, силами свободных вычислительных узлов.

8. Виртуализация среды на базе Copy-on-Write (CoW). Механизм виртуализации файловой системы позволяет тестировать гигантские массивы данных без их копирования на дисках (фигура 9). Система подключает продуктовый массив данных к тестовым серверам в режиме «только чтение».

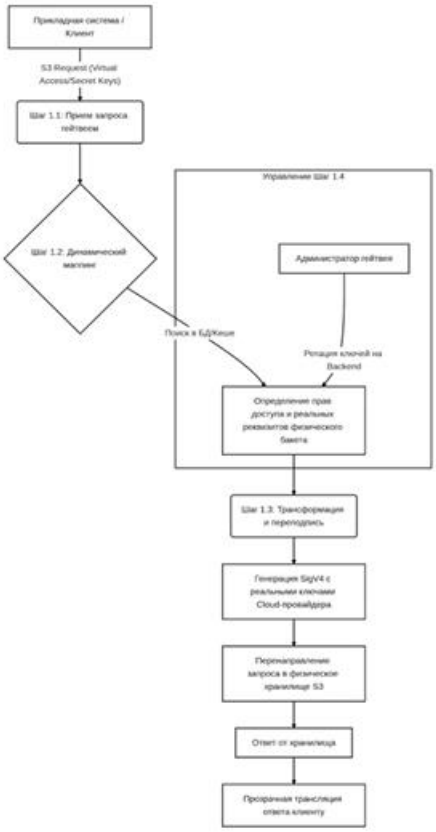
Шаг 8.1: Изоляция изменений через «дельты». Все операции записи сохраняются на обособленный дисковый раздел («дельту»), не затрагивая блоки оригинальных файлов. Шаг 8.2: Динамическая маршрутизация запросов. При обращении к объекту процессор гейтвея проверяет его наличие в слое «дельты» (тестовый контур). Если объект был изменен или создан в рамках теста, система отдает его целиком из хранилища «дельты», в противном случае запрос прозрачно перенаправляется к оригинальному объекту в продуктовом контуре (шаг 8.3).

ФОРМУЛА ИЗОБРЕТЕНИЯ

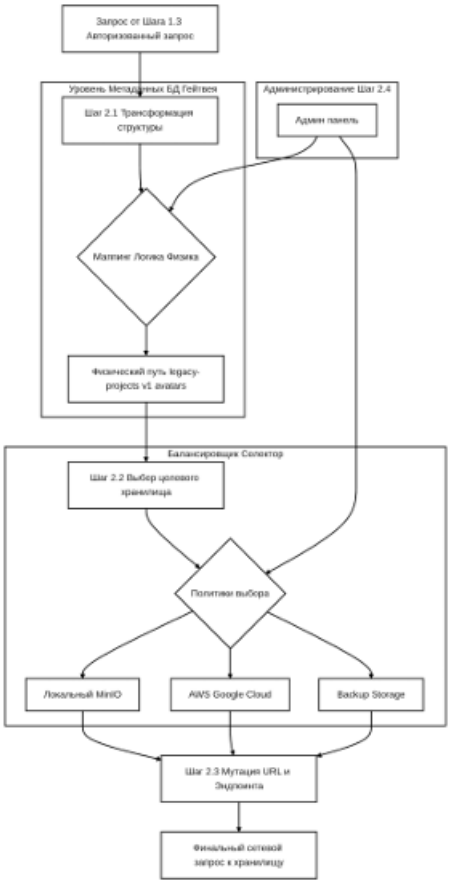
1. Система интеграции прикладных программных систем-потребителей и инфраструктуры хранения данных, содержащая центральную компьютерную станцию с прокси-сервером, выполненным с возможностью приема запросов по протоколу S3, проверки соответствия загружаемых файлов заявленным типам данных и поддержки работы с учетными записями, ***отличающаяся*** тем, что центральная компьютерная станция соединена с сервисом метаданных, вычислительными узлами обработки данных и множеством физически разрозненных S3-хранилищ данных, размещенных в различных центрах обработки данных и/или у различных провайдеров, при этом система содержит: виртуальный S3-агрегатор с динамическим маппингом полномочий, выполненный с возможностью сопоставления виртуальных S3-реквизитов доступа с учетными данными физических хранилищ данных, переподписи запросов и маршрутизации запросов между указанными физическими хранилищами данных; средства виртуализации среды хранения на базе Copy-on-Write, выполненные с возможностью предоставления доступа к данным продуктового контура без физического копирования указанных данных и сохранения изменений в отдельной дельте данных; динамические алиасы, выполненные с возможностью подмены путей, бакетов и структуры размещения объектов при обработке запросов; встроенный фильтр макросов, выполненный с возможностью проверки документов на наличие исполняемых скриптов; масштабируемый вычислительный движок обработки данных, содержащий средства обработки в режиме реального времени, средства асинхронной обработки и средства выполнения заданий для ресурсоемких вычислений; средства исполнения пользовательской логики на языке Python в режиме runtime без перезапуска служб системы; средства глубокого копирования структурированных документов, выполненные с возможностью анализа структуры документа, определения связанных медиа-объектов и физического копирования указанных медиа-объектов; средства отказоустойчивой репликации данных между центрами обработки данных, выполненные с возможностью двусторонней синхронизации данных, применения механизма Skip Logic и восстановления согласованного состояния данных посредством Snapshot-восстановления; при этом сервис метаданных, вычислительные узлы обработки данных, средства репликации данных и физические S3-хранилища образуют единый контур хранения, обработки, репликации и доставки цифровых объектов.

2. Способ интеграции прикладных программных систем-потребителей и инфраструктуры хранения данных, ***характеризующийся*** тем, что: принимают посредством прокси-сервера запрос с использованием виртуальных S3-реквизитов доступа; осуществляют сопоставление виртуальных S3-реквизитов доступа с учетными данными физического хранилища данных; выполняют переподпись запроса, изменяют путь и бакет, а также динамически модифицируют конечную точку запроса; определяют физическое хранилище данных для обработки запроса; выполняют анализ и валидацию загружаемого объекта в оперативной памяти, включая проверку фактического типа объекта на основе анализа магических байтов, выявление ZIP-бомб, JPEG-бомб и анализ макросов; при необходимости выполняют антивирусную проверку объекта; сохраняют объект в физическом хранилище данных; осуществляют обработку объекта посредством вычислительного движка обработки данных в режиме реального времени, асинхронном режиме, режиме задания либо режиме генерации по запросу; при необходимости исполняют пользовательскую логику обработки объекта на языке Python без перезапуска служб системы; при копировании структурированного документа анализируют структуру указанного

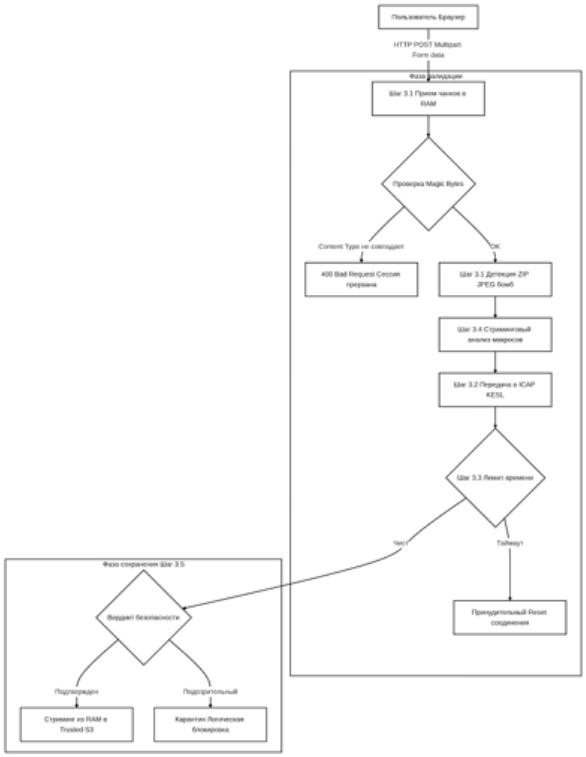
документа, определяют связанные медиа-объекты и выполняют физическое копирование указанных медиа-объектов совместно с документом; осуществляют репликацию данных между центрами обработки данных, при этом при возникновении нарушения связи маркируют объект как проблемный, приостанавливают выполнение повторных операций передачи указанного объекта, а после восстановления связи получают актуальный снимок состояния объекта и синхронизируют данные на основании указанного снимка состояния; предоставляют тестовой среде доступ к данным продуктового контура в режиме только чтения, при этом изменения сохраняют в отдельной дельте данных с использованием механизма Copy-on-Write; осуществляют выдачу объекта прикладной программной системе-потребителю либо пользователю.



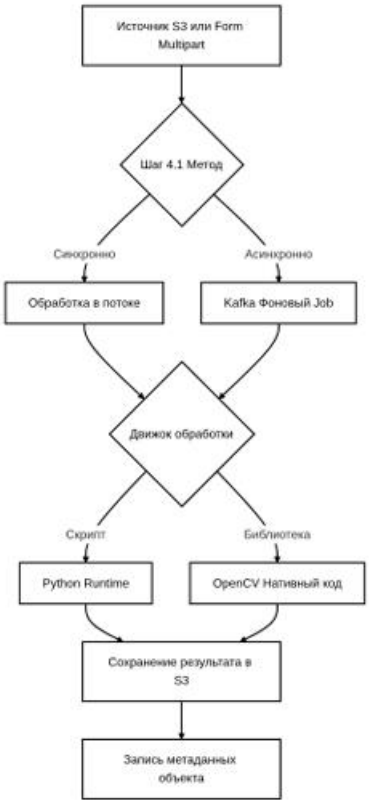
Фигура 1 – Алгоритм авторизации и динамической переподписи S3-запросов через виртуальные реквизиты доступа



Фигура 2 – Механизм логической агрегации распределенных S3-хранилищ с динамической мутацией путей и эндпоинтов



Фигура 3 – Многоуровневый контур превентивной валидации и стримингового анализа безопасности входящего контента



Фигура 4 – Схема конвейерной обработки и деривации контента в синхронном и асинхронном режимах



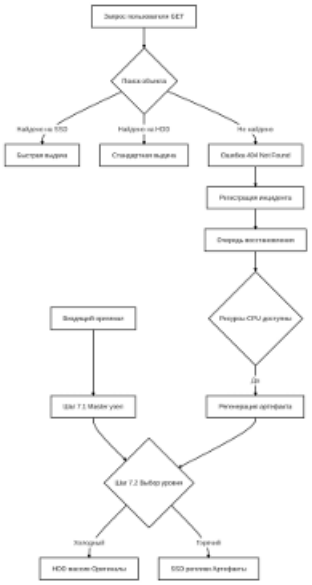
Фигура 5 – Алгоритм адаптивной выдачи контента с использованием каскадного поиска и трансформации «на лету» (On-demand)



Фигура 6 – Процесс рекурсивного клонирования связанных объектов и графа зависимостей Rich Content



Фигура 7 – Метод распределенной отказоустойчивой репликации с разрешением конфликтов версий и оптимизацией трафика



Фигура 8 – Система многоуровневого хранения с самовосстановлением артефактов



Фигура 9 – Технология виртуализации продуктовых данных для изолированных сред на основе механизма перехвата дельт (CoW)

Верстка Д. Женысова
Корректор Г. Косанова