УДК 004.932:004.75

ОПТИМИЗАЦИЯ ОБЛАЧНЫХ ХРАНИЛИЩ ДАННЫХ С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ ДЕДУПЛИКАЦИИ ИЗОБРАЖЕНИЙ

Вегера К.С.

Белорусский государственный университет информатики и радиоэлектроники, г. Минск, Республика Беларусь

Научный руководитель: Марков А.Н. – магистр технических наук, старший преподаватель кафедры информатики

Аннотация. Работа посвящена проблеме избыточности данных в облачных сервисах хранения медиаконтента. Предложена архитектура программного комплекса, реализующего механизм дедупликации изображений на основе перцептивного хеширования (dHash). Описан алгоритм взаимодействия клиентской части, базы данных PostgreSQL и объектного хранилища, позволяющий исключить повторную запись визуально идентичных файлов. Приведена методика оценки эффективности сжатия хранилища.

Ключевые слова: облачные вычисления, дедупликация, архитектура ПО, PostgreSQL, оптимизация ресурсов, dHash.

Введение. Современные IT-инфраструктуры сталкиваются с экспоненциальным ростом объемов неструктурированных данных. По статистике, значительную долю дискового пространства занимают дубликаты изображений [1], возникающие при пересылке контента, резервном копировании или незначительном изменении форматов. Классификация моделей облачных вычислений [2] позволяет выделить специализированные сервисы хранения как критически важный компонент IaaS. Традиционные методы дедупликации, основанные на криптографических хешах (MD5, SHA-256), неэффективны для медиаконтента, так как изменение разрешения или метаданных файла приводит к изменению хеш-суммы, что вынуждает систему хранить идентичные по содержанию копии.

Целью работы является проектирование архитектуры облачного сервиса, обеспечивающего минимизацию затрат на хранение данных (IaaS) за счет выявления и исключения нечетких дубликатов.

Методика и алгоритмы. Для решения задачи предложено использование методики дедупликации на стороне сервера (target-based deduplication) с использованием перцептивных хешей. В качестве базового алгоритма выбран Difference Hash (dHash) [3], который формирует цифровой отпечаток изображения на основе градиентов яркости между соседними пикселями. Процесс генерации хеша dHash состоит из четырех последовательных этапов, каждый из которых направлен на удаление избыточной информации (шума, детализации, цвета):

1. Масштабирование (Resizing). Изображение сжимается до размера 9x8 пикселей (всего 72 пикселя). Использование нечетного количества столбцов (9) необходимо для обеспечения перекрытия при вычислении разницы между соседними точками.

2. Обесцвечивание. Изображение преобразуется в оттенки серого, что позволяет абстрагироваться от цветовой гаммы и сосредоточиться только на структурной информации (форме объектов).

3. Вычисление разницы. Для каждой строки пикселей y вычисляется разница яркости между пикселем x и $x + 1$ пикселем.

4. Построение битовой строки. Если яркость пикселя $P(x) > P(x + 1)$, то бит хеша равен 1, иначе – 0. В результате формируется 64-битное число (8x8), которое является компактным цифровым следом изображения.

Выбор dHash обусловлен его устойчивостью к масштабированию изображения и высокой скоростью генерации (менее 6 мс на изображение), что критически важно для высоконагруженных систем. Критерием идентичности изображений служит расстояние Хэмминга (d_H). Если $d_H \leq 5$, изображения считаются визуальными копиями.

Архитектура системы. Разработанное решение базируется на принципе разделения метаданных и бинарных данных (Blob). Логическая схема хранения реализуется с использованием СУБД PostgreSQL и S3-совместимого хранилища. Процесс загрузки файла включает следующие этапы:

1. Генерация отпечатка. Сервис принимает изображение и вычисляет его перцептивный хеш (64-битное число).

2. Поиск в индексе. Выполняется запрос к таблице images_hash в БД PostgreSQL. Благодаря использованию специализированных индексов (GIST/SP-GIST для битовых строк), поиск похожих хешей выполняется за логарифмическое время. Для хранения 64-битных хешей в PostgreSQL используется тип данных bit(64), который позволяет выполнять побитовые операции непосредственно на уровне СУБД, минимизируя передачу данных между приложением и базой. Расстояние Хэмминга между двумя хешами A и B вычисляется как количество единичных битов в результате операции XOR ($A \oplus B$). Для оптимизации выборки на больших объемах данных (миллионы записей) стандартный B-tree индекс неэффективен. Вместо него предлагается использовать специализированные BK-деревья (Burkhard-Keller trees), которые позволяют эффективно отсекать ветви дерева поиска, где расстояние заведомо превышает заданный порог ($threshold$). Это снижает сложность поиска с линейной $O(N)$ до логарифмической $O(\log_2 N)$.

3. Принятие решения. Сценарий А (дубликат найден): если в БД найден хеш с $d_H \leq treshold$, физическая загрузка файла в S3 отменяется. В таблице user_files создается новая запись, ссылающаяся на уже существующий file_id. Счетчик ссылок (reference count) для оригинала увеличивается. Сценарий Б (уникальный контент): Файл загружается в S3, ему присваивается уникальный ID, а хеш сохраняется в индексе.

Графически алгоритм работы подсистемы представлен на рисунке 1:

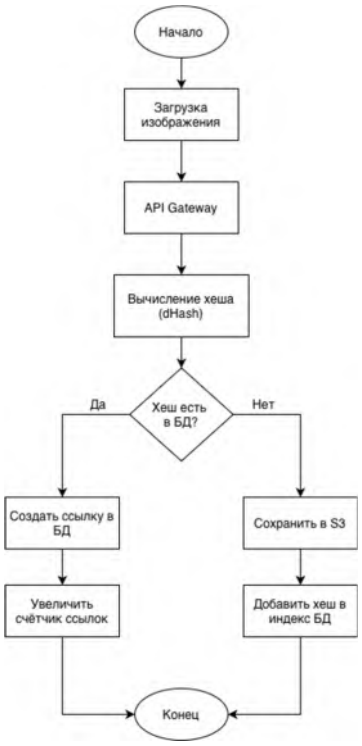

Рисунок 1 – Алгоритм работы подсистемы дедупликации при загрузке данных

Такой подход реализует модель Content-Addressable Storage (CAS), где адресация контента происходит на основе его содержимого, а не имени файла.

Обработка коллизий. При использовании 64-битного хеширования теоретически возможны коллизии, когда разные изображения имеют одинаковый dHash (особенно при сильном размытии или монотонном фоне). Вероятность такого события для фотографий

крайне мала (2^{-64}), однако для повышения надежности системы в архитектуру введен механизм верификации. В случае обнаружения дубликата (расстояние Хэмминга равно 0) система выполняет дополнительную проверку: сравнивает размеры файлов (в байтах) и, при необходимости, вычисляет быстрый криптографический хеш (CRC32) для первых 4 КБ файла. Это позволяет с вероятностью, близкой к 100%, исключить ложные срабатывания (False Positives), не прибегая к полному побитовому сравнению файлов.

Оценка эффективности. Эффективность предлагаемой архитектуры оценивается коэффициентом дедупликации (K_{dedup}), который показывает отношение логического объема данных к физическому, и рассчитывается по формуле 1:

$$K_{dedup} = \frac{V_{logical}}{V_{physical}} = \frac{\sum_{i=1}^{N} Size(f_i)}{Size(Unique) + Size(Meta)}, \quad (1)$$

где $V_{logical}$ – суммарный объем файлов, загруженных пользователями, ГБ;

$V_{physical}$ – реально занимаемое место на дисках, ГБ.

Предварительное моделирование на тестовой выборке из 10 000 изображений (включая ресайзы и пересохранения) показало, что внедрение данного алгоритма позволяет достичь коэффициента $K_{dedup} \approx 1.6$, что эквивалентно экономии 30–45% дискового пространства.

Заключение. Предложена архитектура облачного сервиса хранения, использующая алгоритм dHash для оптимизации использования ресурсов. Переход от прямого хранения файлов к системе ссылок на уникальные контентные единицы позволяет существенно снизить требования к инфраструктуре хранения (IaaS) и уменьшить избыточный сетевой трафик, не снижая качество обслуживания пользователей.

Список литературы

1. *Армбруст М. Облачные вычисления: взгляд сверху // Коммуникации АСМ. – 2010. – Т. 53, № 4.*
2. *Вакеро Л. М. Определение и классификация облачных вычислений // Вестн. информатики. – 2011. – Т. 28.*
3. *Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М.: Техносфера, 2012.*

UDC 004.932:004.75

OPTIMIZATION OF CLOUD DATA STORAGE USING IMAGE DEDUPLICATION ALGORITHMS

Viahera K.S.

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Markov A.N. – Master of Sci. (M. of Sci.), Senior Lecturer at the Department of Informatics

Annotation. The paper considers the problem of data redundancy in cloud media storage services. An architecture of a software complex implementing an image deduplication mechanism based on perceptual hashing (dHash) is proposed. The interaction algorithm between the client part, PostgreSQL database, and object storage, allowing to exclude re-recording of visually identical files, is described. A methodology for assessing storage compression efficiency is presented.

Keywords: cloud computing, deduplication, software architecture, PostgreSQL, resource optimization, dHash.