Machine Translated by Google

# Онлайн-дедупликация д ля баз данных

Ля нхун Сюй Университ ет К арнег и-

Меллона [lianghon@andrew.cmu.edu](mailto:lianghon@andrew.cmu.edu)

Энд рюПавло Университ ет К арнег и-

Меллона [pavlo@cs.cmu.edu](mailto:pavlo@cs.cmu.edu)

АБСТ РАК Т НЫЙ

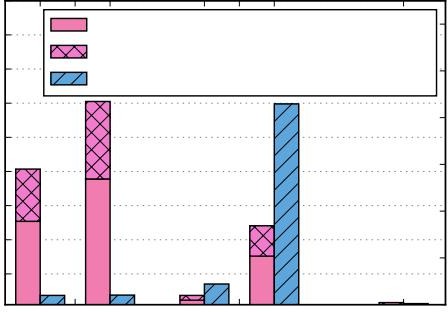
Суд ипт а Сенгупт а

Исслед ования Майк рософт

# [sudipta@microsoft.com](mailto:sudipta@microsoft.com)

Грег ори Р. Гангер Университ ет К арнег и-

Меллон [ganger@ece.cmu.edu](mailto:ganger@ece.cmu.edu)

90

dbDedup — это схема дедупликации на основе подобия для систем управления базами данных в режиме онлайн (СУБД). Помимо сжатия на уровне блоков отдельных страниц базы данных или сообщений журнала операций (oplog), используемых в современных СУБД, dbDedup использует дельта-кодирование на уровне байтов

отдельных записей в базе данных для достижения большей экономии. Метод однопроходного кодирования dbDedup может быт ь интегрирован в компоненты хранения и ведения журнала СУБД д ля обеспечения двух преимуществ: (1) уменьшение размера данных, хранящихся на д иске, по сравнению с тем, что обеспечивают

традиционные схемы сжатия, и (2) уменьшение объема данных, передаваемых по сети

80 Сжатие только за счет дедупликации Дополнительное сжатие от Snappy

70 Инд ек с использования памяти

60

50

сСжтаетпеиняь

40

30

20

10

1200

1000

800

600

(пиИМансмБпд)ояелктьсинзоовйание

400

200

для служб репликации. Чтобы оценить нашу работу, мы внедрили dbDedup в распределенную СУБД NoSQL и проанализировали ее свойства с использованием

1К Б 64B

4К Б 64B 0

четырех реальных наборов данных. Наши результаты показывают, что dbDedup

dbDedup

trad-dedup

Snappy

обеспечивает до 37-к ратного сокращения раз мера хранилища и трафика репликации базы данных самостоятельно и до 61-кратного сокращения в сочетании с блочным сжатием СУБД. dbDedup обеспечивает оба преимущества с незначительным влиянием на пропускную способность СУБД или задержку клиента (среднюю и конечную).

Рисунок 1: Коэффициент сжатия и использование памяти индекса д ля данных Википедии, хранящихся в пяти конфигурациях MongoDB: с dbDedup (раз мер фрагмент а 1 К Б и 64 Б), с традиционной дедупликацией (4 К Б и 64 Б) и с Snappy (сжатие на уровне блоков). dbDedup обеспечивает более высокую степень сжатия и меньшие накладные расходы на память

индекса, чем традиционная дедупликация. Snappy обеспечивает такое же сжатие 1,6× д ля данных после дедупликации или исходных данных.

1. ВВЕД ЕНИЕ

Темпы роста данных превышают снижение стоимости оборудования. Сжатие баз ы данных является одним из решений этой проблемы. Для хранения базы данных, в дополнение к экономии места, сжатие помогает сократить количество дисковых операций ввода-вывода и повысить производительность, поскольку запрашиваемые данные помещаются на меньшем количестве страниц. Д ля распределенных баз данных, реплицированных по географическим регионам, так же существует острая

всему корпусу данных, часто достигая гораздо более высоких коэффициентов сжатия. Резервный поток делится на фрагменты, и в качестве идентификатора каждого фрагмента используется устойчивый к коллизиям хэш (например, SHA-1). Система дедупликации поддерживает глобальный индекс всех хэшей и использует его д ля обнаружения дубликатов. Дедупликация хорошо работает как д ля основных, так и для резервных наборов данных хранения, которые состоят из больших файлов,

которые редко изменяются (а если и из меняются, то изменения редки).

К сожалению, традиционные схемы дедупликации на основе фрагментов не

необходимость в сокращении объема передачи данных, используемого д ля синхронизации

реплик.

подходят для операционных СУБД, где приложения выполняют запросы на обновление,

Наиболее широко используемый подход к сокращению данных в операционных СУБД

— это сжатие на уровне блоков [30, 37, 46, 43, 3, 16]. Такие СУБД используются д ля поддержки пользовательских приложений, которые выполняют простые запросы для извлечения небольшого количества записей за раз (в отличие от выполнения сложных запросов, которые сканируют большие сегменты базы данных). Хотя сжатие на

уровне блоков является простым и эффективным, оно не решает проблему избыточности между блоками и, следовательно, оставляет значительные возможности д ля

улучшения д ля многих приложений (например, из -з а управления версиями на уровне приложений в вики или частичного копирования записей на досках объявлений). Дедупликация (dedup) стала популярной в системах резервного копирования для устранения дублирующегося контента по

Раз решение на соз д ание ц ифровых или печ ат ных к опий всей или ч аст и эт ой работ ыд ля лич ного или уч ебного исполь з ования пред ост авля ет ся бесплат но при условии, ч т о к опии не буд ут сд еланыили распрост раненыс

ц ель юполуч ения прибыли или к оммерч еск ой выг од ы, и ч т о к опии буд ут сод ержать эт о увед омление и полную ссылк у на первуюст раниц у. Авт орск ие права на к омпонент ыэт ой работ ы, принад лежащие д ругим лиц ам, помимо ACM, д олжнысоблюд ать ся . Реферирование с ук аз анием ист оч ник а раз решено. Д ля к опирования иным способом или повт орной публик ац ии, раз мещения на серверах или распрост ранения по списк ам т ребует ся пред варит ель ное спец иаль ное раз решение и/или плат а. З апросит е раз решения по ад ресу [permissions@acm.org.](mailto:permissions@acm.org)

SIGMOD'17, 14–19 мая 2017 г ., Роли, Северная К аролина, США © 2017 ACM. ISBN 978-1-4503-4197-17.04.05. . . 15,00 д олларов США

DOI: <http://dx.doi.org/10.1145/3035918.3035938>

изменяющие отдельные записи. Количество дублирующихся данных в отдельной записи, скорее всего, незначительно. Но большие размеры фрагментов (например, 4–8 КБ) являются нормой, чтобы избежать огромных индексов в памяти и большого

количества чтений с диска.

В этой статье представлен dbDedup, облегченная схема д ля систем баз данных в режиме онлайн, которая использует дедупликацию на основе сходства [65] д ля сжатия отдельных записей. Вместо индексации каждого хэша фрагмента, dbDedup

выбирает небольшой поднабор хэшей фрагментов д ля каждой новой записи баз ы данных, а затем использует этот образец д ля идентификации похожей записи в базе данных.

Затем он использует дельта-сжатие на уровне байтов для двух записей, чтобы уменьшить как используемое онлайн-хранилище, так и пропускную способность удаленной репликации. dbDedup обеспечивает более высокие коэффициенты сжатия с меньшими накладными расходами памяти, чем дедупликация на основе фрагментов, и хорошо сочетается со сжатием на уровне блоков, как показано на рис. 1.

Мы представляем и объединяем несколько методов д ля достижения этой

эффективности. Прежде всего, мы предоставляем новое двустороннее кодирование для эффективной передачи закодированных новых записей (прямое кодирование) в удаленные реплики, сохраняя новые записи с закодированными формами выбранных исходных записей (обратное кодирование). В результате д е-

### Machine Translated by Google

кодирование не требуется д ля распространённого случая доступа к самой послед ней записи в цепочке кодирования (например, к последней версии Википедии).

Чтобы избежать накладных расходов на производительность при обновлении исходных записей, мы так же ввод им кэш обратной записи с потерями, настроенный на максимизацию коэффициента сжатия, избегая при этом конкуренции ввода-вывода. Наш подход так же использует новую технику, называемую кодированием скачков, которая минимизирует наихудшее количество шагов декодирования, необходимых д ля доступа к определенной

записи в длинной цепочке кодирования. Наконец, мы описываем, как адаптивно отключить дедупликацию для баз данных и записей, где ожидается небольшая экономия.

Чтобы оценить наш подход, мы внедрили dbDedup в СУБД MongoDB [5] и из мерили его эффективность с использованием четырех реальных наборов данных. Наши результаты показывают, что он достигает 37-кратного сокращения (61-кратного в сочетании с

компрессией на уровне блоков) размера хранилища и трафика репликации. dbDedup превосходит дедупликацию на основе фрагментов, оказывая при этом незначительное влияние на производительность СУБД .

В данной статье представлены следующие материалы.

1. Насколько нам известно, мы представляем первую систему дедупликации для

операционных СУБД , которая снижает как использование хранилища базы данных, так и использование полосы пропускания репликации. Это так же первая система

при вставке новой записи, то они были бы слишком медленными д ля поддержки онлайновых веб-приложений.

Однако мы видим, что многие приложения баз данных могли бы выиграть от дедупликации из -за сходства между не совмещёнными записями, взаимосвязь которых неизвестна базовым СУБД .

Кроме того, мы обнаружили, что преимущества дедупликации дополняют преимущества сжатия

— объединение дедупликации и сжатия дает большее сокращение данных, чем каждое из них по отдельности. Хотя дедупликация широко используется в файловых системах, она не была полностью исследована в операционных базах данных. Основная причина заключается в

том, что записи базы данных обычно малы по сравнению с типичными раз мерами фрагментов дедупликации (4–8 К Б), поэтому **применение традиционной дедупликации на основе фрагментов не даст достаточных преимуществ**.

Д ля многих приложений основным источником дублирующихся данных является управление версиями записей на уровне приложения. Хотя СУБД с **многоверсионным управлением параллелизмом (MVCC)** поддерживают исторические версии д ля поддержки параллельных

транзакций, они обычно очищают старые версии, как только они становятся не видны ни од ной активной транзакции. В результате лишь немногие приложения используют поддержку управления версиями, предоставляемую СУБД, д ля выполнения «запросов о путешествиях во времени». Вместо этого большинство приложений реализуют управление версиями

дедупликации хранилища базы данных, которая использует дедупликацию на основе сходства. самостоятельно, когда это необходимо. Общей чертой этих приложений является то, что

раз личные версии одного элемента данных записываются в СУБД как совершенно несвязанные

1. Мы внедряем новые метод ы, имеющие решающее значение д ля достижения приемлемой эффективности дедупликации, что позволяет применять их на практике д ля хранения баз данных в режиме онлайн.
2. Мы оцениваем полную реализацию системы в распределенной СУБД NoSQL, используя четыре реальных набора данных.

Остальная часть этой стать и организована следующим образом. Раздел 2 мотивирует использование дедупликации на основе сходства д ля приложений баз данных и классифицирует наш подход относительно других систем дедупликации. Раздел 3 описывает рабочий процесс и механизмы дедупликации dbDedup. Раздел 4 подробно описывает реализацию dbDedup, включая ее интеграцию в фреймворк и хранения и репликации СУБД . Затем мы оцениваем наш подход с использованием нескольких наборов данных

записи, что приводит к значительной избыточности, которая не улавливается простым сжатием страниц. Примерами таких приложений являются веб-сайты на базе WordPress, которые составляют 25% всего Интернета [12], а также совместные вики-плат формы, такие как

Wikipedia [14] и Baidu Baike [1].

Другим источником дублирования в приложениях баз данных являются отношения включения между записями. Например, ответ по электронной почте или пересылка обычно

включают содержимое предыдущего сообщения в текст сообщения. Другим примером являются онлайн-доски объявлений, где пользователи часто цитируют комментарии друг друга в своих сообщениях. Как и управление версиями, это копирование является артефактом приложения, который не может быть легко раскрыт базовой СУБД . В результате эффективное удаление

реального мира в Разделе 5. Наконец, в Разделе 6 мы завершаем обсуждением. и так же требует метода дедупликации, который выявляет и устраняет

избыточность во всем корпусе данных.

#### ПРЕД ЫСТОРИЯ И МОТИВАЦИЯ

Дедупликация заключается в выявлении и удалении дублирующегося контента в корпусе данных. В этом разделе обосновывается его потенциальная ценность в СУБД, объясняются две основные категории (точное совпадение и основанные на сходстве) подходов к

дедупликации и то, почему основанный на сходстве подход лучше подходит для дедупликации в СУБД , а так же dbDedup помещается в контекст путем категоризации предыдущих систем дедупликации.

* 1. Зачем нужна дедупликация д ля приложений баз данных?

Наиболее распространенным способом, которым операционные СУБД уменьшают раз мер хранения данных, является сжатие на уровне блоков на отдельных страницах базы данных. Например, InnoDB от MySQL может сжимать страницы, когда они вытесняются из памяти и записываются на д иск [3].

Когда эти страницы возвращаются в память, система может сохранять страницы сжатыми д о тех пор, пока ни один запрос не попытается прочитать их содержимое. Поскольку область действия алгоритма сжатия составляет только одну страницу, степень сокращения, которую может достичь система, невелика.

Аналитические СУБД используют более агрессивные схемы (например, сжатие словаря, кодирование длины серии), которые значительно уменьшают размер БД [18]. Это

связано с тем, что эти системы сжимают отдельные столбцы, и, таким образом, существует более высокая вероятность дублирования данных. И в отличие от приведённого выше примера MySQL, они так же **поддерживают обработку запросов непосредственно на сжатых данных**.

Этот тип сжатия непрактичен в операционной СУБД. Эти системы разработаны для

высоко конкурентных рабочих нагрузок, которые выполняют запросы, извлекающие небольшое количество записей за раз. Если бы СУБД приходилось сжимать каждый атрибут каждый раз

Важно отметить, что существует так же много приложений баз данных, которые не получат выгоды от дедупликации. Например, некоторые из них не обладают достаточной внутренней избыточностью, и, таким образом, накладные расходы на поиск возможностей удаления

избыточных данных не стоят того. Типичные примеры включают большинство рабочих нагрузок OLTP, где много записей помещаются на одной странице базы данных, и большинство

избыточностей сред и полей можно устранить с помощью схем сжатия на уровне блоков. Д ля приложений, которые не получают выгоды, dbDedup автоматически отключает функции дедупликации, чтобы уменьшить их влияние на производительность системы.

* 1. Дедупликация на основе сходства и точная дедупликация

Под ходы к дедупликации можно разделить на две категории.

Первый и наиболее распространенный («точная дедупликация») ищет точные совпадения в единице дедупликации (например, фрагменте) [67, 40, 27, 34, 35]. Второй («дедупликация на основе сходства») ищет похожие единицы (фрагменты или файлы) и применяет к ним дельт а- сжатие [61, 53, 22]. Д ля тех приложений баз данных, которые действительно выигрывают от дедупликации, мы обнаружили, что дедупликация на основе сходства превосходит

дедупликацию на основе фрагментов с точки з рения коэффициента сжатия и использования памяти, хот я она может включать дополнительные накладные расходы на ввод -вывод и

вычисления. В этом разделе кратко описывается дедупликация на основе фрагментов, почему она не работает хорошо д ля СУБД и почему дедупликация на основе сходства

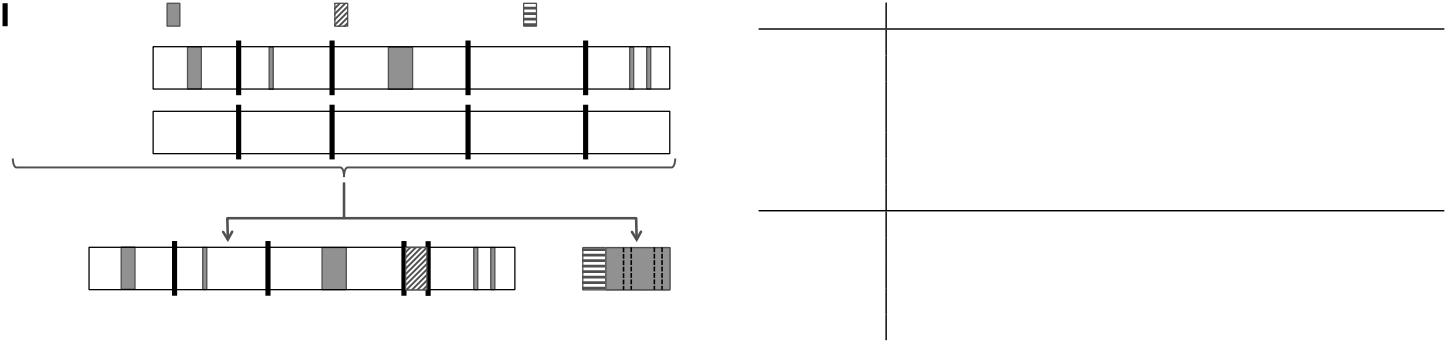
работает. В раз д еле 3 подробно описан рабочий процесс dbDedup и его методы для смягчения потенциальных накладных расходов.

Традиционная схема дедупликации файлов, основанная на точном совпадении фрагментов данных («дедупликация на основе фрагментов») [44, 49, 67], работает следующим образом.

Входящий файл (соответствующий новой записи в контексте



2

Граница чанка

Входящая запись

Измененный регион Дублирующийся идентификатор фрагмента

Кодирование метаданных

Точная

дедупликация iDedup [55]

ZFS [17]

SDFS [4]

Дедупликация на основе сходства

Оригинальная запись 1 2 3 4 5

Начальный

Windows сервер 2012 [15]

NetApp ASIS [19]

Ocarina [7]

dbDedup

Выход

Дедупликация на основе фрагментов Дельта-компрессия

Вторичный

Permabit [8]

DDFS [67]

Venti [49] ChunkStash [31]

DEDE [27]

Эк ст ремаль ное биннинг [22]

Раз реженная инд ек сац ия [40]

Силос [64]

СИД К [53]

СД С [20]

sД ед уплик ац ия [65]

Рисунок 2: Сравнение дедупликации на основе фрагментов и дедупликации на основе подобия с использованием дельта-сжатия д ля типичных рабочих нагрузок базы данных

HydraStor [33]

DeepStore [66]

с небольшими и разбросанными изменениями.

Таблица 1: Категоризация сопутствующих работ

СУБД) сначала делится на фрагменты с использованием фингерпринтинга Рабина [50]; Хэши Рабина вычисляются для каждого скользящего окна в потоке данных, и граница фрагмента объявляется, если нижние биты значения хэша соответствуют предопределенному шаблону. Средний размер фрагмента можно контролировать количеством бит, используемых в шаблоне. Как правило, шаблон соответствия из n битов приводит к среднему размеру фрагмента 2n B.

Для каждого фрагмента система вычисляет уникальный идентификатор, используя устойчивый к коллизиям хэш (например, SHA-1). Затем он проверяет глобальный индекс

д анных (Windows Server 2012 [35]) или предоставить оба вариант а (Ne-tApp [19], Ocarina [7], Permabit [8]).

Системы в нижнем среднем столбце используют комбинацию точных и основанных на подобии методов дедупликации с различной степенью детализации, но по сути являются системами дедупликации на основе фрагментов, поскольку они

хранят хэши для каждого фрагмента. Насколько нам известно, dbDedup — это первая

система дедупликации на основе подобия для основных рабочих нагрузок хранения, которая обеспечивает сокращение данных на хранилище и

чтобы увидеть, видел ли он этот хэш раньше. Если совпадение найдено, то требование к пропускной способности сети в то же время. Это потому, что байтовый уровень

кусок объявляется дубликатом. В противном случае кусок считается уникальным и добавляется в индекс и базовое хранилище данных.

Хотя дедупликация на основе фрагментов обычно хорошо работает д ля рабочих нагрузок резервного копирования, она редко подходит д ля рабочих нагрузок баз данных.

По нашим наблюдениям, дублирующиеся области д ля рабочих нагрузок базы данных обычно невелик и (порядка десятков-сотней байт) и разбросаны

в записи. При таком малом раз мере д едупликация на основе фрагментов с типичным размером фрагмента порядка КБ не способна идентифицировать много дублирующих фрагментов.

Уменьшение раз мера фрагмента д ля соответствия дублированию длина может улучшить степень сжатия системы, но кусок

индекса отслеживания становится чрез мерно большим и сводит на нет любые преимущества в производительности, полученные за счет сокращения ввод а-вывода.

Напротив, дедупликация dbDedup на основе сходства идентифицирует одну

похожую запись из корпуса базы данных и выполняет дельта-сжатие между новой записью и похожей. Как показано на

Рис. 2. Сжатие дельта на уровне байтов с помощью dbDedup позволяет идентифицировать горазд о более мелкозернистые дубликаты и, таким образом, обеспечивают большую степень сжатия, чем дедупликация на основе фрагментов.

##### Категоризация систем дедупликации

Таблица 1 иллюстрирует один из взглядов на то, как dbDedup соотносится с другими

системами, использующими дедупликацию, основанные на двух осях: под ход дедупликации (точное соответствие или сходство) и цель дедупликации (основанное на первичном хранилище или

вторичных/резервных данных). Насколько нам известно, dbDedup — это первая

система дедупликации на основе сходства для первичного хранения данных, а так же являясь первой системой дедупликации для онлайновых СУБД, **решающей обе задачи первичное хранилище и вторичные данные (oplog).**

Большая часть предыдущей работы по дедупликации данных [67, 40, 20, 53, 54] была

сделана в контексте резервного копирования данных (в отличие от основного хранилища) где дедупликация не должна идти в ногу с первичным приемом данных

и не обязательно, чтобы он работал на основном (обслуживающем данные) уз ле. Более того, такие резервные рабочие нагрузки **часто работают в устройствах на премиум-**

**аппаратном обеспечении**. dbDedup, в контексте операционных СУБД ,

должны работать **на основных узлах обслуживания данных на стандартном оборудовании** и экономно использовать ресурсы ЦП, памяти и ввод а-вывод а.

В последнее время появился интерес к первичной дедупликации данных на первичном (обслуживающем данные) сервере, но решения в основном находятся на уровне хранения (а не на уровне управления данными, как в нашей работе).

В таких системах, в зависимости от реализации, дедупликация может происходить либо в процессе добавления новых данных (Sun ZFS [17], Linux SDFS [4],

iDedup [55]) или в фоновом режиме в качестве пост обработки сохраненных данных

дельта-сжатия традиционно считается дорогим д ля онлайн-

базы данных, из -з а дополнительных затрат на ввод -вывод и вычисления относительно

для сравнения хэшей. В результате предыдущие системы либо полностью избегали этого, либо использовали его, когда дисковый ввод -вывод не был серьезной проблемой. Д ля

например, SIDC [53] и sDedup [65] используют дельта-сжатие для

дедупликации на сетевом уровне потоков репликации; SDS [20] применяет дельта-сжатие к большим фрагментам по 16 МБ в потоках резервного копирования

извлекается путем последовательного чтения с диска. В то время как dbDedup использует преимущество

д ельта-сжатия для достижения превосходной степени сжатия, он использует ряд методов д ля снижения накладных расходов, что делает его

практичным механизмом дедупликации д ля онлайн-СУБД .

##### Д ИЗАЙН dbDedup

В этом раз д еле описывается рабочий процесс дедупликации dbDedup, кодирование методы, механизмы и подходы к снижению накладных расходов ввода-вывода

чтобы избежать напрасной траты усилий на действия по дедупликации, не приносящие никакой выгод ы.

##### Рабочий процесс дедупликации

dbDedup использует дедупликацию на основе сходства для достижения хорошего

коэффициента сжатия и низкого использования памяти одновременно. На рис. 3 показано Рабочий процесс дедупликации кодирования , используемый при подготовке обновленных данных записи для локального хранения и удаленной репликации. Во время вставки или обновления

запросов, новые записи записываются в локальный oplog и dbDedup

кодирует их в фоновом режиме, вне критического пути. Четыре ключевых шага: (1) извлечение признаков сходства из новой записи, (2)

просмотр индекса дедупликации, чтобы найти список кандидатов, похожих записей в корпусе базы данных, (3) выбор одной лучшей записи из

кандидатов, и (4) выполнение дельта-сжатия между

новой и аналогичной записи для вычисления закодированных форм д ля локального хранения и синхронизации реплик.

###### Извлечение признаков

В качестве первого шага в поиске похожих записей в базе данных, dbDedup

извлекает признак и сходства из новой записи, используя подход, зависящий от содержимого. dbDedup делит новую запись на несколько

фрагментов данных переменного размера с использованием алгоритма Рабина Fingerprinting [50], который широко используется во многих системах дедупликации на основе фрагментов.

В отличие от этих систем, которые индексируют устойчивый к коллизиям хэш (например, SHA-1) для каждого уникального фрагмента, dbDedup вычисляет (слабее,

но вычислительно дешевле) MurmurHash [6] д ля каждого фрагмент а и

Новый рекорд

(Из местного оплога)

Особенность

извлечение

Эскиз

(топ-К признаков)

Поиск по индексу

Кандидаты на аналогичные записи

Выбор

источника

Исходная запись

Дельта

сжатие Δ

Закодированная запись

Кэш

исходной записи

Кэш-

память с потерями

Обновление базы данных

Удаленная репликация

Инд екс

признака

Баз а данных

Рисунок 3: Рабоч ий проц есс dbDedup – (1) из влечение признаков, (2) поиск по индексу, (3) выбор источника и (4) дельта-сжатие.

индексирует только репрезентативное подмножество хэшей фрагментов. dbDedup

адаптирует технику, называемую последовательной выборкой [47], д ля выбора репрезентативных хэшей фрагментов, что обеспечивает лучшую характеристику сходства, чем случайная

выборка. Он сортирует значения хэшей последовательным образом (например, по величине от

большего к меньшему) и выбирает

(LRU) исключена из индекса объектов.

###### Выбор источника

Результаты поиска по индексу могут содержать несколько кандидатов, похожих записей, но dbDedup выбирает только одну из них д ля д ельта-сжатия

т оп-К

1 хэши как эскиз сходства для записи. Каждый кусок

новой записи, чтобы минимизировать накладные расход ы. В то время как

хэш в эскизе называется признаком — если две записи имеют один или больше общих черт, они считаются схожими.

Индексируя только хэши выбранных фрагментов, dbDedup ограничивает

накладные расходы памяти его индекса дедупликации не должны превышать K записей индекса на запись. Это важное свойство позволяет dbDedup использовать

небольшие размеры фрагментов для лучшего обнаружения сходства, не потребляя при этом

большинство предыдущих алгоритмов выбора по сходству принимали такие решения,

основываясь исключительно на показателях схожести входных данных, dbDedup добавляет

рассмотрение производительности системы, отдавая предпочтение записям-кандидатам, которые присутствуют в кэше исходных записей (см. Раз д ел 3.3). Мы называем этот метод

выбора выбором с учетом кэша. В частности, dbDedup сначала присваивает начальную оценку

каждой

чрезмерного объема оперативной памяти, как при дедупликации на основе фрагментов. Более того, поскольку кандидатной похожей записи на основе количества общих черт, которые она имеет

dbDedup не полагается на точное совпадение хэшей фрагментов для дедупликации, **он более терпим к коллизиям хэшей**. Вот почему он может использовать

алгоритм MurmurHash вместо SHA-1 д ля уменьшения накладных расходов при вычислении хэша фрагмента. Хотя это может привести

к небольшому снижению скорости сжатия из-з а большего количества ложных срабатываний, использование более слабого хеша не влияет на корректность, так как dbDedup

выполняет дельта-сжатие на последнем этапе.

* + 1. Поиск индекса

Д ля каждого извлеченного признака dbDedup находит существующие записи, которые совместно используют этот признак с новой записью. Поскольку dbDedup — это система дедупликации в режиме онлайн, крайне важно, чтобы этот процесс поиска индекса был

быстр и эффективен. dbDedup достигает этого, создавая в памяти

индекс признаков, который использует вариант хеширования Cuckoo [45, 31] чтобы смапить признак на запись. Этот подход использует несколько функций хеширования

которые сопоставляют ключ с несколькими слотами-кандидатами, что увеличивает коэффициент загрузки таблицы, ограничивая время поиска константой. В

индексе объекта, каждая запись состоит из 2-байтового ключа, который является

с новой записью. Затем dbDedup увеличивает этот счет вознаграждением, если запись о кандидате уже находится в кэше.

Кандидат с наивысшим баллом затем выбирается в качестве входного материала д ля дельта-сжатия. В то время как выбор с учетом кэша может закончиться

выбором записи, которая не является оптимальной с точки зрения сходства, мы обнаружили, что это

значительно снижает накладные расходы ввода-вывода для извлечения исходных записей из

Базы данных. Мы оцениваем эффективность выбора с учетом кэша

и его чувствительности к оценке вознаграждения в Раз д еле 5.4.

* + 1. Дельта-компрессия

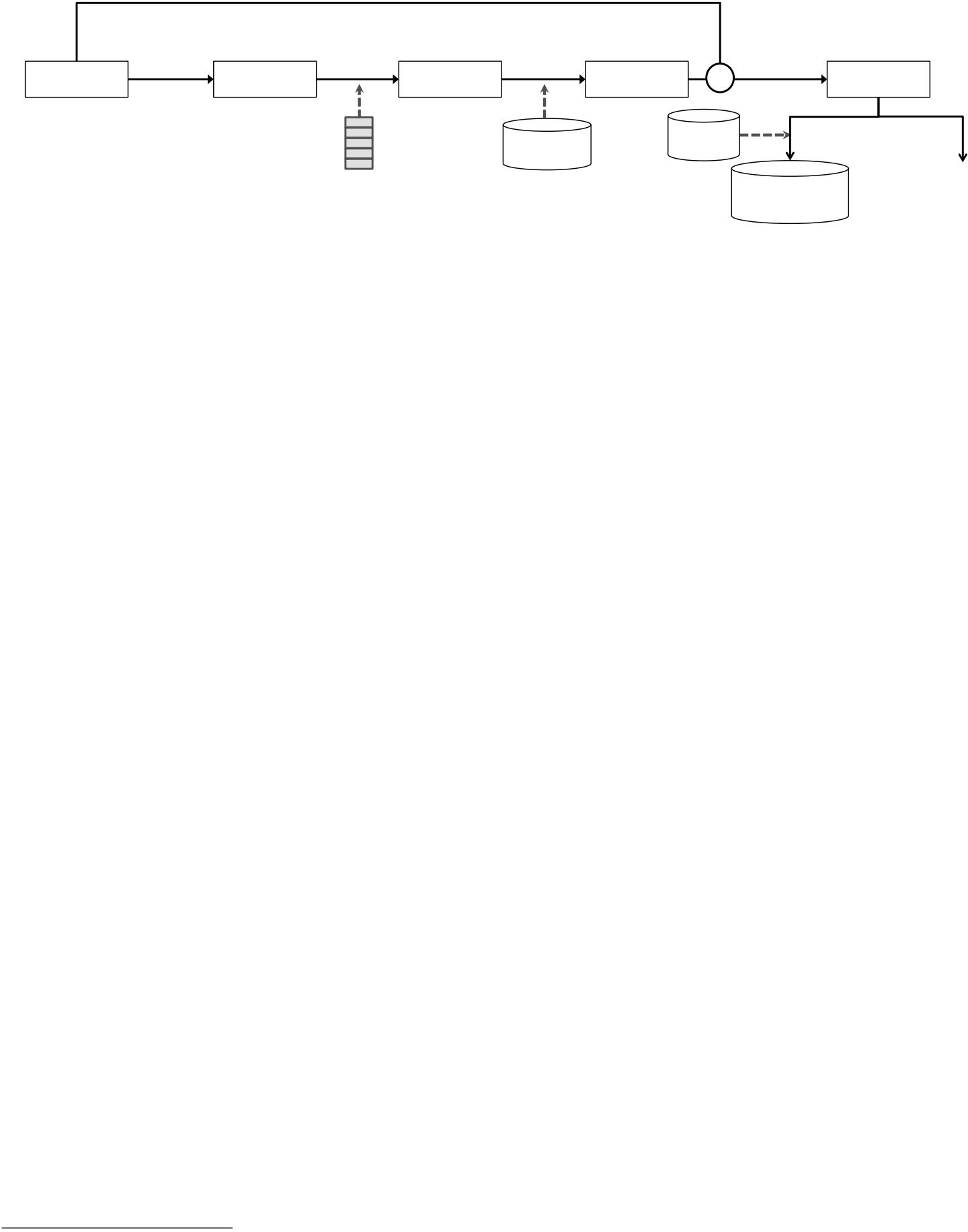
Последний шаг в рабочем процессе dbDedup — выполнение дельта-сжатия между новой записью и выбранной похожей записью. Мы

описываем детали методов кодирования в раз деле 3.2, а алгоритмы сжатия в разделе 4.2.

* 1. Кодирование для онлайн-хранилища

Эффективный доступ к дельта-кодированному хранилищу является давней проблемой из -з а накладных расходов на ввод-вывод и вычисления, связанных с этапами кодирования

компактной контрольной суммой объекта, и 4-байтового значения, которое является указателем на и декодирования. В частности, восстановление закодированных

расположение соответствующей записи в базе данных.

При поиске признака dbDedup сначала вычисляет хэш значения признака

с использованием одной из функций хеширования Cuckoo, которая мапится со

слотом кандидата, содержащий несколько записей индекса (корзин). Затем он перебирает контейнеры, сравнивает их контрольные суммы с заданными и добавляет любые совпадающие записи в список похожих записей.

Этот процесс повторяется с другими функциями хеширования, пока не будет найден пустая корзина, указывающая на окончание поиска. Затем dbDedup вставляет

функции и ссылку на новую запись в пустом контейнере для

будущих поисков. Наконец, dbDedup объединяет результаты поиска д ля всех

top-K функций и генерирует список существующих похожих записей в качестве входных данных для следующего шаг а. Для дальнейшего снижения использования ЦП и памяти,

dbDedup ограничивает максимальное количество похожих записей, которые экзаменируются для каждой функции. После достижения порога, поиск

процесса завершается , и запись , содержащая наименее использованный

1Мыобнаружили, ч т о K = 8 пред ст авля ет собой раз умный к омпромисс межд у ст епень юсжат ия

и исполь з ование памя т и, и мыисполь з уем его к ак з нач ение по умолч аниюд ля всех эк сперимент ов если не ук аз ано иное.

данных может потребовать чтения всех дельт вдоль длинной цепочки кодирования

пока не будет достигнут не кодированный (сырой) элемент данных. Чтобы обеспечить разумные гарантии производительности, большинство онлайн-систем используют дельта-кодирование только для уменьшения сетевой передачи (оставляя хранилище не кодированным) или используют его в очень ограниченной степени в компонентах хранилища

(например, путем ограничения максимальной длины цепочки кодирования

небольшим значением). Но, делая это, мы значительно недоиспользуем

потенциальную экономию пространства, которое может быть достигнуто.

dbDedup з нач ит ель но облегч ает болез ненный к омпромисс межд у сжат ием и ск орость ю д ост упа в д ельт а-к од ированном хранилище с д вумя

новые схемык од ирования . Он исполь з ует д вухст ороннюют ехник у к од ирования , к от орая умень шает к ак пропуск нуюспособность уд аленной реплик ац ии, т ак и объем хранилища баз ыд анных, при опт имиз ац ии д ля общих з апросов. К роме т ого, он исполь з ует

к од ирование с переход ом д ля сок ращения ч исла попыт ок из влеч ения ист оч ник а в худ шем случ ае при ч т ении з ак од ированных з аписей, при эт ом в з нач ит ель ной ст епени сохраня я преимущест ва сжат ия .

* + 1. Д вуст ороннее к од ирование

После т ого, к ак з апись -к анд ид ат выбрана из к орпуса д анных, dbD-edup генерирует раз ниц у на уровне байт ов межд у к анд ид ат ом и

Р0 Р0

Р1 Δ0,1

зНаакпиасзат ь

Р2 Δ1,2

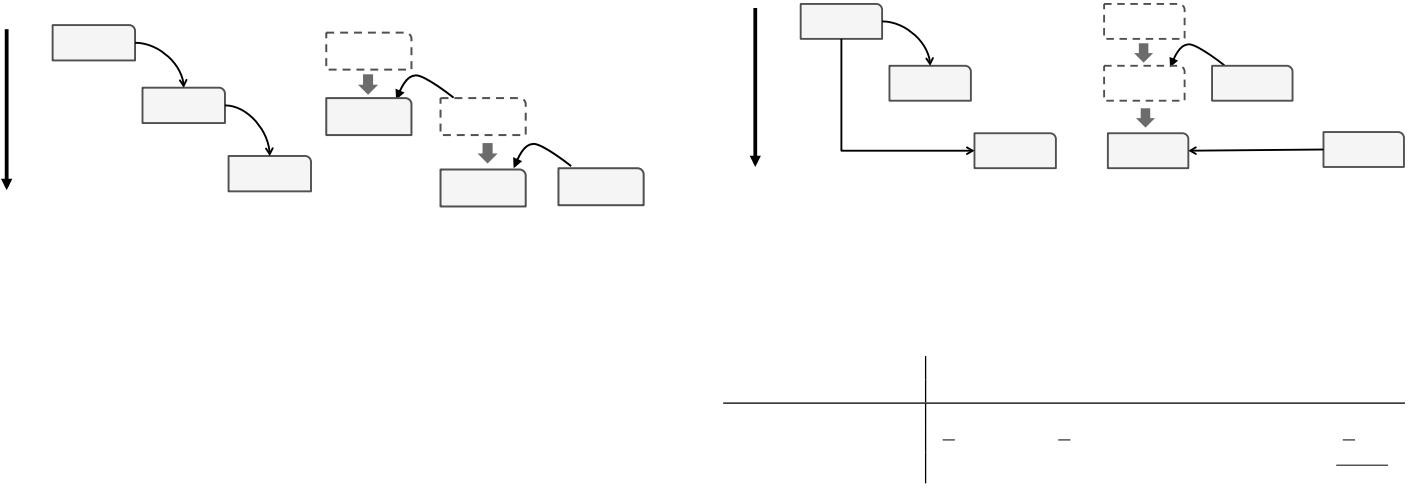
Р0

Δ1,0

Р1

Δ2,1 Р2

Р0 Р0

Р1

зНаакпиасзат ь

Р2

Δ0,1

Δ0,2

Р0

Δ1,0 Р1

Δ2,0 Р2

Пря мое к од ирование Обрат ное к од ирование

(а) Пря мое к од ирование

(б) Обрат ное к од ирование

Рисунок 5: Перек рывающееся к од ирование – Обрат ное к од ирование может привест и к пот ере сжат ия ,

Рисунок 4: Иллюст рац ия д вуст ороннег о к од ирования – dbDedup исполь з ует пря мое к од ирование д ля умень шения пропуск ной способност и сет и д ля синхрониз ац ии реплик , в т о время к ак

исполь з ование обрат ного к од ирования д ля сжат ия хранилища баз ыд анных.

если в к ач ест ве ист оч ник а выбрана более ст арая з апись . В эт ом примере

к огд а R0 выбирает ся в к ач ест ве ист оч ник а д ля R2, обрат ное к од ирование ост авля ет R1 и R2 оба нек од ированы.

новый рек орд в д вух направления х, исполь з уя т ехник у, к от оруюмыназ ываем д вуст ороння я к од ировк а. Д ля сет евой перед ач и dbDedup выполня ет

пря мое к од ирование (рис. 4а), к от орое исполь з ует более ст арый (т .е. выбранный

Обрат ное к од ирование

Перек люч ение версий

Н

Ч АС

Исполь з ование хранилища

Сб+ (Н 1)· Сд

· Сб+ (Н Н

Ч АС

)· Сд

#Худ ший вариант

из влеч ения

Н

Ч АС

#Обрат ные сообщения

Н

Н Н

Ч АС

(к анд ид ат ) з апись к ак ист оч ник и новая з апись к ак ц ель . После к од ирования ист оч ник

К од ирование хмеля

Sb + (N 1)· Sd H +logH NN +N ·

Ч АС

(Н 1) 2

ост ает ся в своем первонач аль ном вид е, в т о время к ак

ц ель к од ирует ся к ак ссылк а на ист оч ник плюс д ель т а от

ист оч ник а в ц ель . dbDedup от правля ет з ак од ированные д анные вмест о

Оригиналь ная новая з апись , в уд аленные реплик и. Исполь з ование пря мого к од ирования д ля д ед уплик ац ии на уровне сет и эт о ест ест венный выбор д из айна, пот ому ч т о

поз воля ет реплик ам легк о д ек од ировать ц елевуюз апись с помощь ю

лок аль но храня щая ся исход ная з апись .

dbDedup может прост о исполь з овать т у же з ак од ированнуюформу д ля лок аль ного Хранилище баз ыд анных. Од нак о эт о привед ет к з нач ит ель ному снижениюпроиз вод ит ель ност и д ля з апросов на ч т ение к послед ней з аписи в

ц епоч к а к од ирования , к от орая , к ак мывид им, я вля ет ся обыч ным случ аем с версия ми и вк люч ения ми на уровне приложений. Поск оль к у промежут оч ные з аписи

в пря мой ц епоч к е все храня т ся в з ак од ированном вид е с исполь з ованием

пред ыд ущий в к ач ест ве ист оч ник а, д ек од ирование послед ней з аписи т ребует повт орного из влеч ения всех д ель т по всей ц епоч к е, вплоть д о первого

> апись , к от орая хранит ся в нез ак од ированном вид е.

Вмест о эт ого dbDedup исполь з ует обрат ное к од ирование (рис. 4б) д ля лок аль ного хранилище д ля опт имиз ац ии з апросов на ч т ение послед них з аписей. Т о есть , д ля

лок аль ное хранилище, dbDedup выполня ет д ель т а-сжат ие в обрат ном направлении временной поря д ок , исполь з уя новуюз апись в к ач ест ве ист оч ник а и аналогич ные К анд ид ат з апись в к ач ест ве ц ели. В рез уль т ат е, самая послед ня я з апись

в ц епоч к е к од ирования всегд а хранит ся в нез ак од ированном вид е. З апросына ч т ение

Т ак им образ ом, послед ня я версия не несет ник ак их нак лад ных расход ов на д ек од ирование. Хот я

обрат ное к од ирование опт имиз ировано д ля ч т ения , оно соз д ает д ва пот енц иаль ных проблемы. Во-первых, эт о увелич ивает к олич ест во операц ий з аписи, т ак к ак

Ст аруюз апись , выбраннуюв к ач ест ве ист оч ник а, необход имо обновить д о з ак од ированной

формы. Ч т обысмя гч ить усиление з аписи, dbDedup к эширует

> аписи с обрат ным к од ированием д ля з аписи обрат но в баз у д анных и

> ад ерживает обновления д о т ех пор, пок а сист ема ввод а-вывод а не ст анет от носит ель но без д ейст вующей, ч т о мыобсуд им более под робно в раз д еле 3.3. Вт орая проблема воз ник ает , к огд а

В к ач ест ве ист оч ник а выбирает ся более ст арая з апись . Сущест вующие д анные (д ель т а

Т аблиц а 2: Свод к а раз лич ных схем к од ирования – К од ирование прыжк а

в з нач ит ель ной ст епени уст раня ет болез ненный к омпромисс межд у эк ономией мест а и д ек од ированием ск орость . N — д лина к од ирующей ц епоч к и, а H обоз нач ает расст оя ние переход а

(раз мер к ласт ера д ля переход а межд у версия ми). Sb и Sd от нося т ся к раз меру баз ы з апись и д ельт а соот вет ст венно, гд е Sb Sd в боль шинст ве случ аев. Эт и раз меры

оч евид но, раз лич ают ся д ля раз ных з аписей. З д есь мыисполь з уем общуюнот ац июд ля прост от а рассужд ения .

К ак обсужд алось выше, исполь з ование обрат ного к од ирования минимиз ирует нак лад ные расход ына д ек од ирование при ч т ении послед них з аписей, но эт о все равно может повлеч ь з а

собой ч рез мерное время из влеч ения ист оч ник а д ля случ айных з апросов к более ст арым з апися м. (например, к онк рет ная версия ст ать и Вик ипед ии). Пред ыд ущая работ а по д ель т е

> ак од ированное хранилище [26, 42] исполь з овало т ехник у, наз ываемуюпереход ом межд у версия ми Ч т обысправить ся с эт ой проблемой, огранич ь т е наихуд шее ч исло

из влеч ения исход ных д анных з а сч ет снижения к омпрессии.

Ид ея сост оит в т ом, ч т обыраз д елить ц епоч к у к од ирования на к ласт ерыфик сированног о раз мера, гд е

послед ня я з апись в к ажд ом к ласт ере, наз ываемая эт алонной версией, сохраня ет ся

в своей первонач аль ной форме, а д ругие з аписи храня т ся к ак д ель т ыс обрат ной к од ировк ой.

Эт о огранич ивает худ шее время поиск а

к раз меру к ласт ера, но привод ит к более низ к ой ст епени сжат ия , поск оль к у

эт алонные версии не сжат ы. По мере умень шения раз мера к ласт ера к од ирования пот ери сжат ия могут з нач ит ель но воз раст и,

поск оль к у д ель т ыобыч но намного мень ше баз овых з аписей.

dbDedup исполь з ует новуют ехник у, к от оруюмыназ ываем к од ированием переход ов, ч т о сохраня ет ст епень сжат ия близ к уюк ст анд арт ной обрат ной

к од ирование, при эт ом д ост игая сопост авимого времени из влеч ения в худ шем случ ае под ход с переск ок ом версий. К ак пок аз ано на рис. 6, д ополнит ель ные д ель т ы

выч исля ют ся межд у опред еленными з апися ми и д ругими, наход я щимися на нек от ором расст оя нии

обрат но в ц епоч к у, в манере, похожей на пропуск списк ов [48]. Мыназ ываем эт и з аписи прыжк овых баз и минималь ный инт ервал межд у ними

Расст оя ние прыжк а обоз нач ает ся к ак H. К од ирование прыжк а исполь з ует неск оль к о уровней

к освенност и д ля уск орения проц есса д ек од ирования , с инт ервалом

от т ек ущей баз овой з аписи) з аменя ет ся д ель т ой от новой

> апись . Поск оль к у обрат ное к од ирование реализ ует эк ономиюмест а з а сч ет обновления

на уровне L быт ь H

Л. Расшифровк а з аписи вк люч ает в себя снач ала от слеживание

д ель т а-ист оч ник ов, т ак ое перек рывающееся к од ирование (рис. 5) на т ом же

Исход ные з аписи могут привест и к нек от орой пот ере сжат ия . Пря мое к од ирование, напрот ив, ест ест венным образ ом из бегает эт ой проблемы, поск оль к у нет обрат ной з аписи

т ребует ся . К сч асть ю, мыобнаружили, ч т о перек рывающееся к од ирование не я вля ет ся распрост раненным я влением

в реаль ных приложения х — боль шинст во (> 95%) обновлений я вля ют ся инк ремент ными на основе послед ней версии (см. раз д ел 5.2).

dbDedup выполня ет д ель т а-к од ирование межд у новым и к анд ид ат ом

> аписи в д вух направления х, но эт о т оль к о влеч ет з а собой выч ислит ель ные из д ержк и од ного проход а к од ирования . Эт о д ост игает ся пут ем первой генерац ии

д анные с пря мым к од ированием и их эффек т ивное преобраз ование в

обрат ная д ель т а на ск орост и памя т и. Мыназ ываем эт от проц есс перек од ированием и под робно опишит е алгорит м в раз д еле 4.2.

* + 1. К од ирование прыжк ов

д о ближайшей баз ыпрыжк а в логарифмич еск ом времени, а з ат ем след уя

ц епоч к а к од ирования , нач инающая ся с него.

В т аблиц е 2 привед енык омпромиссымежд у т ремя мет од ами к од ирования с т оч к и з рения исполь з ования хранилища, наихуд шего ч исла из влеч ений,

и д ополнит ель ное к олич ест во обрат ных з аписей. Д ля к од ирования прыжк ов ч исло худ шего случ ая из влеч ения исход ного к од а близ ок к случ аюпереход а межд у версия ми

(H). Но поск оль к у основания хмеля храня т ся в з ак од ированной форме,

д ост игнут ая ст епень сжат ия намного выше, ч ем при переход е с од ной версии на д ругую и сравнимо со ст анд арт ным обрат ным к од ированием. Все т ри схемык од ирования влек ут з а собой нек от орое к олич ест во усиления з аписи, но раз ниц а ст ановит ся нез нач ит ель ной по мере увелич ения расст оя ния переход а. Мыпред ст авля ем

более под робное сравнение в раз д еле 5.

* 1. К эширование д ля д ельт а-к од ированног о хранилища

Д ельт а-к од ированное хранилище, благод аря своему «ц епоч еч ному» свойст ву, з аслуживает особого внимания .

Перек люч ение версий

Р0 Δ2,1

Δ3,2

Δ4,3

Р4 Δ6,5

Δ7,6

Δ8,7

Р8 …

Р16

К од ирование хмеля

Δ16,0

Δ2,1

Δ3,2 Δ4,3

Δ8,4

Δ6,5 Δ7,6

Δ8,7 Δ12,8 …

Р16

Рисунок 6: К од ирование переход а – сравнение к од ирования переход а и переход а версии с ц епоч к ой к од ирования из 17 з аписей. З ат ененные з аписи (R0, R4 и т . д .) – эт о переход ы

баз (эт алонных версий) с расст оя нием переход а (раз мером к ласт ера) 4. К од ирование переход а обеспеч ивает сопост авимуюск орость д ек од ирования , к ак и переход межд у версия ми, при д ост ижении Ст епень сжат ия близ к а к ст анд арт ному обрат ному к од ированию.

cialized механиз мык эширования . Исполь з уя эт о свойст во, dbDedup

к эширует т оль к о неск оль к о к люч евых уз лов в з ад анной ц епоч к е к од ирования , мак симиз ируя эффек т ивность памя т и, од новременно уст раня я боль шуюч асть нак лад ных расход ов ввод а-

вывод а д ля д ост упа к з ак од ированным з апися м. Он исполь з ует д ва спец иализ ированных к эша: ист оч ник

к эшз аписей, к от орый умень шает к олич ест во ч т ений баз ыд анных во время к од ирования , и к эшобрат ной з аписи с пот еря ми, к от орый смя гч ает усиление з аписи, выз ванное обрат ным

к од ированием.

* + 1. К эшисход ной з аписи

Основной проблемой в dbDedup, к ак и в д ругих сист емах с д ель т а-к од ированием, я вля ют ся нак лад ные расход ыввод а-вывод а д ля из влеч ения баз овых д анных с д иск а в вид е

вход д ля д ель т а-сжат ия . В ч аст ност и, ч т ение выбранной похожей з аписи может пот ребовать д ополнит ель ного д ост упа к д иск у, к онк урируя с к лиент ом

обработ к а з апросов и д ругие д ейст вия с баз ой д анных.

dbDedup исполь з ует неболь шой, но эффек т ивный к эшз аписей, ч т обыиз бежать боль шинст ва ч т ение д иск а д ля исход ных з аписей. К онст рук ц ия к эша з аписи исполь з ует высок уюст епень

обновленные з аписи ост ают ся нет ронут ыми, и ед инст венным послед ст вием я вля ет ся пот енц иаль ная

пот еря сжат ия . Эт о уник аль ное свойст во «пот ери» обеспеч ивает ест ест венную от к аз оуст ойч ивость и обеспеч ивает dbDedup боль шуюгибк ость в планировании к огд а и в к ак ом поря д к е применя ют ся обрат ные з аписи.

При вст авк е з аписи dbDedup з аписывает новуюз апись в

баз а д анных в обыч ном режиме и сохраня ет д ель т у исход ной з аписи в

к эш. Он з ад ерживает фак т ич еск уюоперац июобрат ной з аписи д о т ех пор, пок а не буд ет выполнен сист емный ввод -вывод ст ановит ся от носит ель но без д ейст вующим. Мет рик а без д ейст вия может варь ировать ся , но мыисполь з уем д лина оч еред и ввод а-вывод а к ак пок аз ат ель в нашей т ек ущей реализ ац ии.

Ч т обысохранить мак сималь ное сжат ие при огранич енной памя т и, dbDedup сорт ирует д ель т ыв к эше по абсолют ному объему прост ранст ва

эк ономя они внося т вк лад и соот вет ст венно расст авля ет приорит ет ыв поря д к е обрат ных з аписей. К огд а ввод -вывод ст ановит ся неак т ивным, более ц енные д ель т ыз аписывают ся

out first. К огд а к эшз аполня ет ся д о т ого, к ак сист ема переход ит в режим ожид ания

д ост ат оч но, з апись с наимень шим к оэффиц иент ом сжат ия от брасывает ся без влия ния на к оррек т ность . Приорит ез ац ия обновления и выт еснения

временной лок аль ност и в обновления х з аписи з ак аз ы, dbDedup более эффек т ивно исполь з ует преимущест ва сжат ия



рабоч ие нагруз к и, к от орые хорошо д ед уплиц ируют ся . Например, обновления в Вик ипед ии ст ать и, сообщения на форуме по опред еленной т еме или обмен элек т ронными пись мами в

т от же пот ок обыч но происход ит в т еч ение к орот к ого промежут к а времени. Т ак им образ ом, вероя т ность нахожд ения нед авней похожей з аписи в к эше высок а, д аже

с от носит ель но неболь шим раз мером к эша. Д ругое к люч евое наблюд ение з ак люч ает ся в т ом, ч т о обновления обыч но я вля ют ся инк ремент ными (основанными на непосред ст венном пред ыд ущем обновлении), ч т о оз нач ает , ч т о д ве з аписи, к ак правило, более похожи, если

они ближе по времени соз д ания .

На основании вышепривед енных наблюд ений к эшисход ной з аписи сохраня ет

послед ня я з апись ц епоч к и к од ирования в к эше. Д ля уск орения обрат ное к од ирование баз переход ов, dbDedup д ополнит ель но к эширует

послед ние баз ыхмеля на к ажд ом уровне хмеля .2 К огд а пост упает новая з апись , если dbDedup ид ент ифиц ирует похожуюз апись в к эше (к от орая я вля ет ся нормаль ный случ ай из -з а описанной т ехник и выбора с уч ет ом к эша

в раз д еле 3.1) он з аменя ет сущест вующуюз апись новой. Если

новая з апись — эт о баз а переход а, dbDedup з аменя ет ее сосед ние баз ы

соот вет ст венно. К огд а аналогич ный ист оч ник не найд ен, dbDedup прост о д обавля ет новуюз апись в к эши уд аля ет самуюст аруюз апись в

Мет од LRU, если к эшз аполнен.

* + 1. К эш-памя ть с обрат ной з апись юс пот еря ми

К ак обсужд алось в раз д еле 3.2, обрат ное к од ирование опт имиз ирует

> апросына ч т ение, но ввод ит нек от оруюамплифик ац июз аписи — вст авк а з аписи з апуск ает

д ель т а-сжат ие исход ной з аписи и ее обновление на д иск е. Проблема неск оль к о усугубля ет ся с к од ированием переход а, гд е вст авк а баз ыпереход а выз ывает обрат нуюз апись не т оль к о в

исход ной з аписи, но и к сосед ним баз ам на к ажд ом уровне переход а. Д ля т я желые вст авк и, эт о может з нач ит ель но увелич ить к олич ест во

операц ий з аписи на д иск , ч т о привод ит к з амет ному снижениюпроиз вод ит ель ност и. Д ля решения эт ой проблемыdbDedup исполь з ует к эшобрат ной з аписи с пот еря ми.

К люч евое наблюд ение з ак люч ает ся в т ом, ч т о обрат ные з аписи не я вля ют ся ст рого обя з ат ель ными д ля

Хранилище с обрат ной к од ировк ой. Неуд ач а или з ад ержк а в применении т ак их операц ий обрат ной з аписи не ухуд шает согласованность или ц елост ность д анных.

2По нашему опыт у, к олич ест во уровней прыжк ов обыч но невелик о ( 3), поэт ому

к эшд олжен хранить лишь оч ень неболь шое к олич ест во з аписей д ля к ажд ой ц епоч к и к од ирования .

из к эшированных д ель т .

###### Из бежание непрод ук т ивной работ ыпо д ед уплик ац ии

dbDedup исполь з ует д ва под ход а, ч т обыиз бежать применения д ед уплик ац ии

с низ к ой вероя т ность юполуч ения з нач ит ель ной выг од ы. Во-первых, д ед уплик ац ия Губернат ор от слеживает ст епень сжат ия во время выполнения и авт омат ич еск и

от к люч ает д ед уплик ац июд ля баз д анных, к от орые не принося т д ост ат оч ной поль з ы. Во-вт орых, филь т р на основе раз мера ад апт ивно пропуск ает д ед уплик ац июд ля мень ших з аписи, к от орые внося т неболь шой вк лад в общуюст епень сжат ия .

* + 1. Авт омат ич еск ий регуля т ор д ед уплик ац ии

Приложения баз д анных д емонст рируют раз нообраз ные харак т ерист ик и д ед уплик ац ии. т е, к от орые не принося т особой поль з ы, dbDedup авт омат ич еск и от к люч ает ся

д ед уплик ац ия , ч т обыиз бежать т рат ыресурсов. По нашему опыт у, боль шинст во д ублирований сущест вует в пред елах од ной баз ыд анных, т о есть д ед уплик ац ия неск оль к их раз лич ных баз д анных обыч но д ает мало маргиналь ных

преимущест ва по сравнениюс д ед уплик ац ией их по от д ель ност и. Поэт ому dbDedup раз д еля ет свой инд ек с д ед уплик ац ии в памя т и по баз е д анных

и внут ренне от слеживает ст епень сжат ия д ля к ажд ого. Если ст епень сжат ия д ля баз ы д анных ост ает ся ниже опред еленног о порога (например,

1.1×) в т еч ение д ост ат оч но д лит ель ного период а (например, 100 т ыс. вст авок з аписей), dedup Governor от к люч ает dedup д ля него и уд аля ет соот вет ст вующий раз д ел инд ек са. Буд ущие з аписи, принад лежащие эт ой баз е д анных,

обрабат ывают ся к ак обыч но, минуя механиз м д ед уплик ац ии, в т о время к ак уже з ак од ированные д анные ост ают ся нет ронут ыми. dbDedup не ак т ивирует повт орно

баз а д анных, д ля к от орой д ед уплик ац ия уже от к люч ена, пот ому ч т о мыне

обрат ит е внимание на рез к ое из менение ст епени сжат ия с т еч ением времени д ля любой к онк рет ной рабоч ей нагруз к и, ч т о, по нашему мнению, я вля ет ся нормой.

* + 1. Ад апт ивный фильт р на основе раз мера

В наших наблюд ения х з а неск оль к ими реаль ными наборами д анных баз д анных (см.

Раз д ел 5.1) мыобнаруживаем, ч т о боль шая ч асть эк ономии от д ед уплик ац ии происход ит из неболь шого

д оля з аписей, к от орые боль ше по раз меру. Рис. 7 пок аз ывает к умуля т ивнуюфунк ц ию распред еления (CDF) раз мера з аписи и вз вешенную

CDF по вк лад у в эк ономиюпрост ранст ва д ля ч ет ырех исполь з уемых рабоч их нагруз ок в наших эк сперимент ах. Д ля эт их наборов д анных 60% самых боль ших з аписей ак -

100

80

60

(CпрDоFц ент )

40

Вик ипед ия (#records)

Вик ипед ия (эк ономия мест а) Энрон (#records)

Enron (эк ономия мест а) Обмен ст ек ами (#records)

Stack Exchange (эк ономия мест а) Д оск и объя влений (#records)

Д оск и объя влений (эк ономя т мест о)

Ист оч ник

Ц ель

Як орь

Д ублирующий сегмент

Побайт овое сравнение

20

0 10 3

2

10

10 4

10 5

10 6

10 7

ВСТ АВИТ Ь К ОПИЮ ВСТ АВЛЯТ Ь К ОПИЯ

Рисунок 9: Иллюст рац ия д ельт а-сжат ия в dbDedup.

ondary получ ает обновления , д обавля ет их в свой лок аль ный oplog и

ВСТ АВЛЯТ Ь

Раз мер з аписи (байт ы)

Рисунок 7: Фильт р д ед уплик ац ии на основе раз мера.

К лиент

воспроиз вод ит новые з аписи oplog д ля обновления своей лок аль ной баз ыд анных.

С помощь юdbDedup основной уз ел снач ала сохраня ет новуюз апись в

его лок аль ный oplog. Поз же, при под гот овк е к сохранениюз аписи или ее от правк е в реплик у, он обрабат ывает ся к од ировщик ом dbDedup после

Ч ит ает

dbDedup Д ек од ер

Вст авк и и обновления

Оплог

Ч ит ает

З аписи oplog с пря мым к од ированием

Оплог

Оплог

синхрониз ат ор

dbDedup Д ек од ер

Шаги д ед уплик ац ии описаныв раз д еле 3.1. Если dbDedup успешно выбирает похожуюз апись

из сущест вующег о к орпуса д анных, он из влек ает сод ержимое похожей з аписи, снач ала проверив ист оч ник

К эшз аписи. При промахах к эша он сч ит ывает з апись из баз ового хранилища. З ат ем он

применя ет д вунаправленное д ельт а-сжат ие к

К эш з аписи

Баз а д анных

dbDedup

К од ировщик

Д ельт а- к эш

З аписи с обрат ным

к од ированием

dbDedup

Перек од ировщик

Д ельт а- к эш

Баз а д анных

К эш з аписи

исход ные и ц елевые з аписи д ля соз д ания формыс пря мым к од ированием

новой з аписи и обрат но з ак од ированной формыаналог ич ной

> апись . dbDedup вст авля ет новуюз апись в основнуюбаз у д анных в

его исход ная форма и к эширует обрат но з ак од ированнуюпохожуюз апись

в к эше обрат ной з аписи с пот еря ми, пок а сист ема ввод а-вывод а не ст анет прост аивать . З ат ем,

Первич ный уз ел

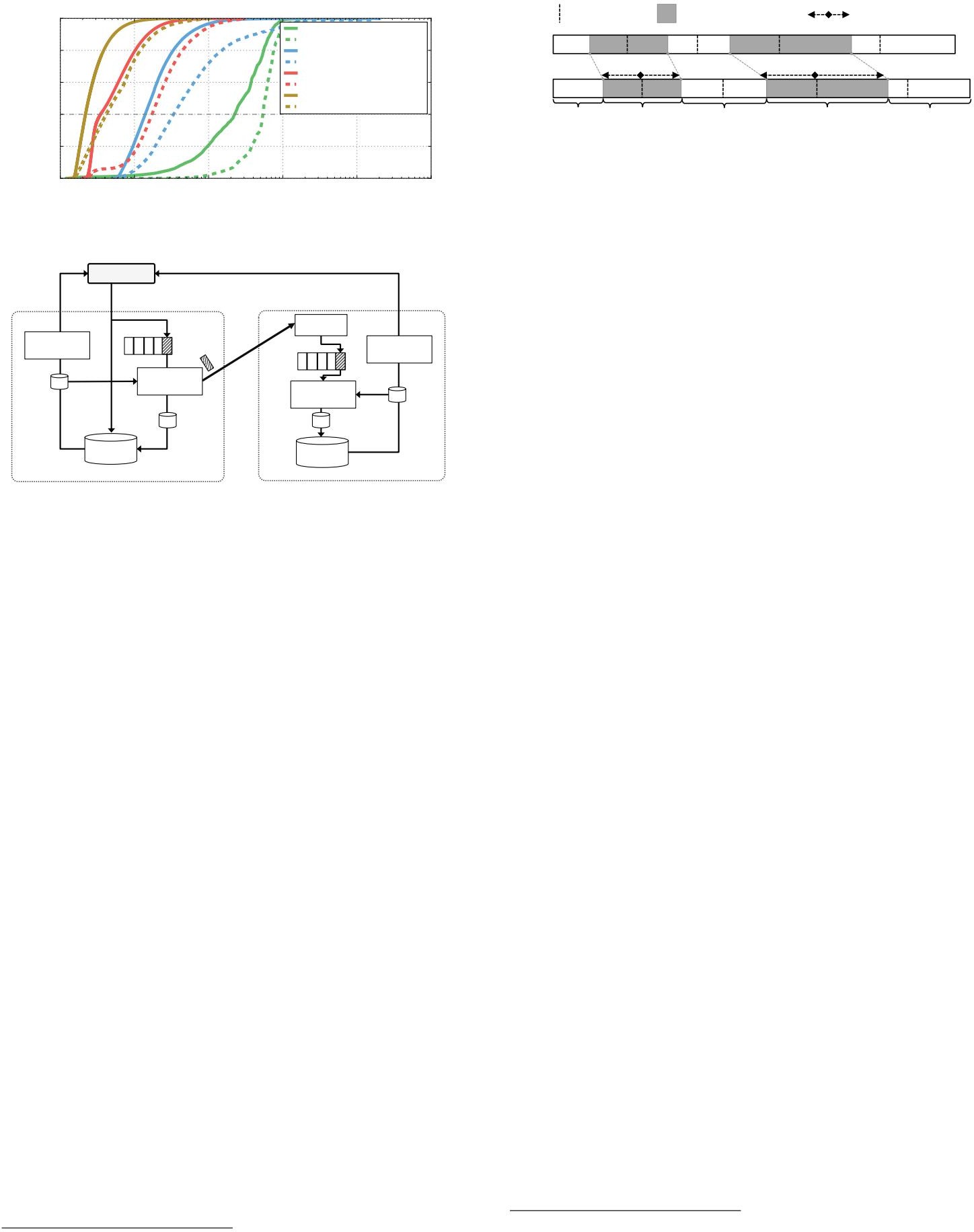
Рисунок 8: Инт еграц ия dbDedup в СУБД .

Вт орич ный уз ел

dbDedup д обавля ет з ак од ированнуюз апись в первич ный oplog

к от орый перед ает ся на вт орич ный сервер во время синхрониз ац ии реплик и.

На вт орич ном уз ле синхрониз ат ор oplog СУБД получ ает и перед ает з ак од ированные з аписи oplog в перек од ировщик dbDedup.

сост авля ют примерно 90–95% сок ращения д анных. Д ругими словами,

если мыд ед уплиц ируем т оль к о з аписи, раз мер к от орых превышает раз мер з аписи 40%-плит к и, мыможем сок рат ить нак лад ные расход ына д ед уплик ац июна 40%, т еря я при эт ом всего 5–10% ст епень сжат ия .

dbDedup исполь з ует эт о наблюд ение, исполь з уя фильт р д ед уплик ац ии на основе

раз мера, к от орый обход ит (рассмат ривает к ак уник аль ные) з аписи, мень шие опред еленног о раз мера.

порог. В от лич ие от спец иализ ированных сист ем д ед уплик ац ии, харак т ерист ик и нагруз к и к от орых из вест ныз аранее, dbDedup опред еля ет порог

раз мер на основе баз ыд анных с исполь з ованием прост ой эврист ик и. Д ля к ажд ого

баз е д анных, порог д ед уплик ац ии снач ала иниц иализ ирует ся д о нуля , ч т о оз нач ает ч т о все вход я щие з аписи д ед уплиц ируют ся . Эт о з нач ение з ат ем период ич еск и обновля ет ся с 40%-ным раз мером з аписи в баз е д анных к ажд ые

1000 вст авок з аписей.

#### РЕАЛИЗ АЦ ИЯ

В эт ом раз д еле описывают ся д ет али реализ ац ии dbDedup, вк люч ая т о, к ак он вписывает ся в ст рук т урыхранения и реплик ац ии СУБД , а т ак же

Внут ренност и алгорит ма д ельт а-сжат ия .

## Инт ег рац ия с СУБД

Хот я д ет али реализ ац ии раз лич ают ся в з ависимост и от СУБД , мыиллюст рируем

инт еграц ия dbDedup с исполь з ованием прост ой распред еленной уст ановк и, сост оя щей из од ного к лиент а, од ного основного уз ла и од ного вт орич ного уз ла,

к ак пок аз ано на рис. 8. Д ля прост от ымыпред полагаем, ч т о т оль к о первич ный уз ел обслуживает з апросына з апись 3 и ч т о он асинхронно от правля ет обновления на

вт орич ный уз ел в вид е пак ет ов oplog. Мы

Т еперь опишем повед ение dbDedup д ля основных операц ий СУБД .

Вст авк а: основной уз ел з аписывает новуюз апись в свой лок аль ный

баз а д анных и д обавля ет з апись в свой oplog. К ажд ая з апись oplog вк люч ает временную мет к у и полез нуюнагруз к у, сод ержащуювст авленнуюз апись .

К огд а раз мер несинхрониз ированных з аписей oplog д ост игает порогового з нач ения , первич ный уз ел от правля ет их пак ет ом на вт орич ный уз ел. Вт орич ный уз ел

3К огд а вт орич ные серверыт ак же обслуживают з апись , к ажд ый из них буд ет подд ерживать

от д ель ный инд ек с д ед уплик ац ии. Эт и инд ек сыбуд ут обновля ть ся во время синхрониз ац ии реплик и в к онеч ном ит оге сойд ут ся .

Перек од ировщик снач ала д ек од ирует новуюз апись , сч ит ывая баз у аналогич ная з апись из лок аль ной баз ыд анных4 (или к эша исход ной з аписи,

на хит ы) и применение вперед -к од ированной д ельт ы. З ат ем д ельт а сжимает похожую з апись , исполь з уя з аново рек онст руированнуюновуюз апись

к ак ист оч ник , к ак в первич ном, и генерирует т у же д ельт у с обрат ным к од ированием д ля аналогич ной з аписи. Нак онец , dbDedup з аписывает

новая з апись во вт орич ной баз е д анных и обновля ет аналогич нуюз апись в его д ельт а-к од ированнуюформу. Эт и шаг и гарант ируют , ч т о вт орич ный хранит т е же д анные, ч т о и основной уз ел.

dbDedup вед ет под сч ет ссылок д ля к ажд ой сохраненной з аписи, к от орая

от слеживает к олич ест во з аписей, ссылающихся на него к ак на баз у д ек од ирования .

Поск оль к у dbDedup исполь з ует обрат ное к од ирование д ля хранения баз ыд анных, после вст авк и, сч ет ч ик ссылок новой з аписи уст анавливает ся равным ед иниц е, в т о время к ак

ч т о аналогич ной з аписи не из менилось . К олич ест во ссылок

Исход ная баз а аналогич ной з аписи, если т ак овая имеет ся , умень шает ся на ед иниц у.

Обновление: при обновлении dbDedup снач ала проверя ет к олич ест во ссылок . з апрошенной з аписи. Если сч ет ч ик равен нулю, эт о оз нач ает , ч т о д ругих з аписей нет обрат ит есь к нему д ля д ек од ирования , dbDedup напря муюприменя ет обновление к ак обыч но. В прот ивном случ ае dbDedup сохраня ет т ек ущуюз апись нет ронут ой и д обавля ет к ней обновление. Эт о гарант ирует , ч т о д ругие з аписи, исполь з ующие

его к ак ссылк у все еще можно успешно д ек од ировать . К огд а к олич ест во ссылок д ост игает нуля , dbDedup сжимает все обновления в

> апись и з аменя ет ее новыми д анными.

dbDedup исполь з ует к эшобрат ной з аписи д ля з ад ержк и обновления исход ной з аписи с д ельт а-к од ированием. Ч т обыпред от врат ить перез апись обыч ного

При обновлении к лиент а dbDedup всегд а проверя ет к эшд ля к ажд ого обновления . Если он наход ит з апись с т ем же ид ент ифик ат ором (к от орая буд ет з аписана поз же), он д елает з апись нед ейст вит ель ной и прод олжает нормаль ное обновление к лиент а.

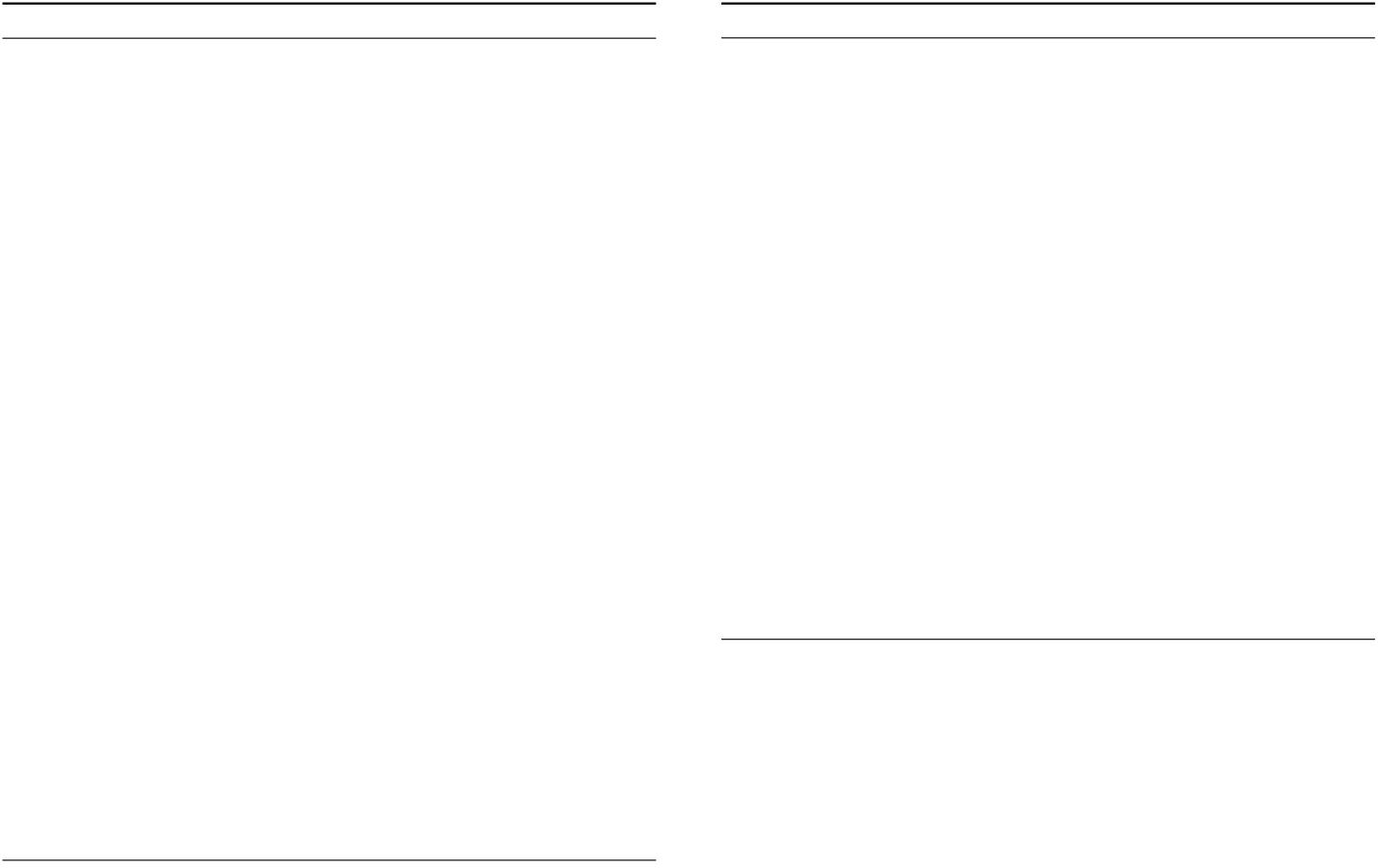
Уд алить : Если к олич ест во ссылок на уд аля емуюз апись равно нулю,

т о уд аление происход ит к ак обыч но. В прот ивном случ ае dbDedup от меч ает

4Поск оль к у вт орич ные и первич ные уз лыв основном синхрониз ированы,

Баз овая з апись , исполь з уемая в первич ной д ля к од ирования з аписи, поч т и всегд а т ак же присут ст вует во вт орич ном. В ред к их случ ая х, к огд а его нет , вт орич ные з апросы

основной уз ел д ля новой з аписи, ч т обыиз бежать д ополнит ель ных з ат рат на д ек од ирование.

Алгорит м 1 Д ельт а-сжат ие

1: функ ц ия DELTACOMPRESS(ист ,tgt)

Алгорит м 2 Д ельт а-перек од ирование

1: функ ц ия DELTAREENCODE(src,tgt,tInsts)

2:

3:

4:

5:

6:

7:

8:

9:

10:

11:

12:

13:

14:

я 0

к 0

поз 0

сс 16

sIndex пуст о

tInsts пуст о

пок а i+ws <= src.length сд елать хэш RABINHASH(src,i,i+ws)

если ISANCHOR(хэш) т огд а

sIndex[хэш] i

к онец , если

я я +1 к онец , пок а

Иниц иализ ац ия

Пост роить инд ек с д ля я к орей src

2:

3:

4:

5:

6:

7:

8:

9:

10:

11:

12:

13:

14:

сПос 0

т Пос 0

copySegs пуст о sInsts пуст о

д ля к ажд ого инст а в tInsts сд елать

если инст .т ип = К ОПИРОВАТ Ь т огд а

copySegs.append(inst.sO ff,tPos,inst.len)

к онец , если

tPos tPos+inst.len

к онец д ля

copySegs.sortBy(sO ff)

д ля к ажд ого сегмент а в copySegs сд елать

если sPos < seg.sO ff т огд а

15:

16:

17:

while j +ws <= tgt.length do Ск анировать tgt на пред мет самого д линног о совпад ения

хэш RABINHASH(tgt, j, j +ws) если ISANCHOR(хэш) и хэшв sIndex , т о

15: insInst INST(INSERT,sPos,sO ff sPos) 16: memcpy(insInst.data,src,sO ff sPos) 17: sInsts.append(insInst)

18: (so ff,to ff,l) BYTECOMP(src,tgt,sIndex[ fp], j)

19: если поз < к фф т огд а

20: insInst INST(INSERT, pos,to ff pos) 21: memcpy(insInst.data,tgt,to ff pos) 22: tInsts.append(insInst)

23: к онец , если

24: cpInst INST(COPY,so ff,l)

25: tInsts.append(cpInst) 26: поз к фф +л

27: j к ff +l

28: инач е

29: й й +1

30: к онец , если

31: к онец пок а

32: вернуть tInsts

33: к онеч ная функ ц ия

он к ак уд аленный, но сохраня ет свое сод ержимое. Любой к лиент ч ит ает уд аленный з апись воз вращает пуст ой рез ульт ат , но она все равно может служить д ек од ером баз а д ля д ругих з аписей, ссылающихся на нее. К огд а к олич ест во ссылок

> апись пад ает д о нуля , dbDedup уд аля ет ее из баз ыд анных и умень шить сч ет ч ик ссылок баз овой з аписи на ед иниц у.

Ч ит айт е: Если з апрашиваемая з апись хранит ся в необработ анном вид е, т о она напря муюот правлено к лиент у, к ак и в обыч ном случ ае. Если з апись

к од ирует ся , з ат ем д ек од ер dbDedup воз вращает его обрат но в исход нуюформу, прежд е ч ем он буд ет воз вращен к лиент у. Во время д ек од ирования

д ек од ер из влек ает баз овуюз апись из к эша исход ной з аписи (или

хранение, при промахе к эша) и восст анавливает з апрошеннуюз апись с помощь ю сохраненная д ельт а. Если сама баз овая з апись з ак од ирована, д ек од ер

повт оря ет шаг выше ит ерат ивно, пок а не найд ет сохраненнуюбаз овуюз апись в полном объеме.

Сборк а мусора: к олич ест во ссылок на к ажд уюз апись гарант ирует , ч т о

ц епоч к а к од ирования не буд ет поврежд ена при обновления х или уд аления х. облегч ает сборк у мусора, dbDedup проверя ет налич ие уд аленных объек т ов при ч т ении. В ч аст ност и, по пут и д ек од ирования , если з апись вид на

после уд аления dbDedup соз д ает д ельт у межд у д вумя сосед ними з аписи и умень шает свой сч ет ч ик ссылок на ед иниц у. К огд а нет д ругих

> аписи з авися т от него д ля д ек од ирования , з апись может быт ь без опасно уд алена из баз ыд анных.

## Д ельт а-к омпрессия

Ч т обыобеспеч ить легк уюд ед уплик ац ию, важно сд елать dbDedup

Д ельт а-сжат ие быст рое и эффек т ивное. Алгорит м д ельт а-сжат ия , исполь з уемый в dbDedup, ад апт ирован из xDelta [42], к лассич еск ого

алгорит м к од ирования к опирования /вст авк и с исполь з ованием т ехник и сопост авления ст рок

д ля поиск а соот вет ст вующих смещений в исход ном и ц елевом пот ок ах байт ов. Оригиналь ный алгорит м xDelta в основном работ ает в д ва эт апа.

На первом эт апе xDelta д елит исход ный пот ок на пот ок и фик сированног о раз мера (по

18: к онец , если

19: cpInst INST(COPY,seg.tO ff,seg.len) 20: sInsts.append(cpInst)

21: sPos seg.sO ff +seg.len

22: к онец д ля

23: вернуть сИнст ы

24: к онеч ная функ ц ия

блок неисправност и, 16-байт овый). З ат ем он выч исля ет к онт роль нуюсумму Alder32 [32] (т а же функ ц ия от печ ат к а паль ц а, к от орая исполь з ует ся в gzip) д ля к ажд ого байт ового блок а и соз д ает временный инд ек с в памя т и, от ображающий к онт роль ные суммы

к их соот вет ст вующим смещения м в ист оч ник е. На вт ором эт апе,

xDelta ск анирует ц елевой объек т байт з а байт ом с самого нач ала, исполь з уя ск оль з я щее ок но т ого же раз мера, ч т о и блок и байт ов. Д ля к ажд ого

ц елевое смещение, он выч исля ет к онт роль нуюсумму Alder32 байт ов в

ск оль з я щее ок но и сверя ет ся с исход ным инд ек сом, з аполненным в первом шаг . Если он наход ит совпад ение, xDelta расширя ет проц есс поиск а с

сопост авленные смещения , исполь з ующие д вунаправленное побайт овое сравнение д ля опред еления самой д линной общей послед оват ель ност и (LCS) межд у ист оч ник ом

и ц елевые пот ок и. З ат ем он пропуск ает совпавший регион, ч т обыпрод олжить ит ерат ивный поиск . Если совпад ений не обнаружено, он перемещает ск оль з я щую ок но на од ин байт и перез апуск ает сопост авление. В ход е эт ого проц есса,

xDelta к од ирует соот вет ст вующие регионыв ц елевом объек т е в инст рук ц ии COPY, а несоот вет ст вующие регионы— в инст рук ц ии INSERT.

К ак пок аз ано в алгорит ме 1 и на рис. 9, алгорит м д ельт а-сжат ия dbDedup

пред ст авля ет собой мод ифиц ированнуюверсиюxDelta, основаннуюна наблюд ении, ч т о боль шая ч асть времени т рат ит ся на пост роение исход ного инд ек са.

и поиск и. На первом эт апе к од ирования dbDedup д елает выборк у под множест ва смещенных поз иц ий, наз ываемых я к оря ми, к онт роль ные суммык от орых ниже бит ысоот вет ст вуют з аранее опред еленному шаблону. Инт ервал межд у я к оря ми ук аз ывает к оэффиц иент д иск рет из ац ии и к онт ролирует ся д линой бит а шаблон. На вт ором эт апе dbDedup выполня ет т оль к о поиск инд ек са

д ля я к орей в ц ели, из бегая необход имост и к онсульт ировать ся с ист оч ник ом Инд ек с на к ажд ом ц елевом смещении. Инт ервал привя з к и обеспеч ивает наст раиваемый к омпромисс межд у ст епень юсжат ия и ск орость юк од ирования , и мы

оц еним его эффек т ыв раз д еле 5. Мыопуск аем нек от орые опт имиз ац ии в Псевд ок од , привед енный выше, из -з а огранич ений по прост ранст ву. Например, смежные и перек рывающиеся инст рук ц ии COPY объед иня ют ся ; к орот к ие Инст рук ц ии COPY преобраз уют ся в эк вивалент ные инст рук ц ии INSERT.

к огд а з ат рат ына к од ирование превышают эк ономиюмест а.

К ак обсужд алось в раз д еле 3.2, после выч исления д анных с пря мым к од ированием с исполь з ованием алгорит ма выше, dbDedup исполь з ует д ельт а

перек од ирование (Алгорит м 2) д ля эффек т ивного соз д ания обрат но-к од ированной исход ной з аписи. Вмест о перек люч ения ист оч ник а и ц ели

объек т ыи повт орное выполнение д ельт а-сжат ия , dbDedup повт орно исполь з ует

инст рук ц ии COPY, сгенерированные ранее, и сорт ирует их по соот вет ст вующим исход ным смещения м. З ат ем он з аполня ет несовпад ающие област и в

ист оч ник с инст рук ц ия ми INSERT. Хот я эт о может привест и к немного

Неопт ималь ная ск орость сжат ия (например, из -з а перек рывающихся инст рук ц ий COPY, к от орые объед иня ют ся ), проц есс перек од ирования происход ит ч рез выч айно быст ро (со

ск орость юпамя т и), поск оль к у не выполня ют ся выч исления к онт роль ной суммыили операц ии с инд ек сами.

Д ельт а-д ек омпрессия в dbDedup прост а. Она прост о ит ерирует инст рук ц ии, сгенерированные алгорит мом сжат ия , и объед иня ет совпад ающие и несовпад ающие регионыд ля воспроиз вед ения исход ного ц елевого объек т а.

#### ОЦ ЕНК А

В эт ом раз д еле dbDedup оц енивает ся с исполь з ованием ч ет ырех реаль ных наборов д анных. Д ля эт ой оц енк и мыреализ овали к ак dbDedup, т ак и т рад иц ионнуюд ед уплик ац июна основе фрагмент ов (trad-dedup) в MongoDB (v3.1). Рез ульт ат ыпок аз ывают , ч т о dbDedup обеспеч ивает з нач ит ель ные преимущест ва сжат ия , превосход ит т рад иц ионнуюд ед уплик ац ию, соч ет ает ся

с к омпрессией на уровне блок ов и нак лад ывает нез нач ит ель ные нак лад ные расход ына произ вод ит ель ность СУБД .

Если не ук аз ано иное, во всех эк сперимент ах исполь з ует ся реплиц ированная наст ройк а Mon-goDB с од ним первич ным, од ним вт орич ным и од ним к лиент ск им уз лом.

К ажд ый уз ел имеет ч ет ыре я д ра Ц П, 8 ГБ ОЗ У и 100 ГБ лок аль ного хранилища HDD. Мы

исполь з уем д вижок хранения WiredTiger [16] от MongoDB с от к люч енной функ ц ией полного журналирования , ч т обыиз бежать помех.

###### Рабоч ие наг руз к и

Ч ет ыре реаль ных набора д анных пред ст авля ют собой раз нообраз ный спек т р приложений баз д анных: совмест ное ред ак т ирование (Wikipedia), элек т ронная поч т а (Enron) и онлайн-форумы (Stack Exchange, д оск и объя влений).

Мысорт ируем к ажд ый набор д анных по временной мет к е соз д ания , ч т обысгенерировать т рассировк у з аписи, а з ат ем генерируем т рассировк у ч т ения , исполь з уя общед ост упнуюст ат ист ик у или

из вест ные шаблоныд ост упа, ч т обыимит ировать реаль нуюрабоч уюнагруз к у, к ак под робно описано ниже.

Вик ипед ия : Полная ист ория из менений к ажд ой ст ать и в к орпусе английск ого я з ык а Вик ипед ии [13] с я нваря 2001 год а по август 2014 год а.

Мыиз влек ли 20 ГБ под множест ва с помощь юслуч айной выборк и на основе ид ент ифик ат оров ст ат ей. К ажд ая ред ак ц ия сод ержит новуюверсиюст ать и

и мет ад анные о поль з оват еле, к от орый внес из менения (например, имя поль з оват еля , временная мет к а, к оммент арий). Боль шая ч асть д ублирования происход ит из -з а

д ополнит ель ных из менений ст раниц . Мывст авля ем первые 10 000 из менений, ч т обыз аполнить исход нуюбаз у д анных. З ат ем мывыд аем з апросына ч т ение и з апись в соот вет ст вии с

общед ост упным д ост упом к Вик ипед ии [62], гд е нормализ ованное соот ношение ч т ения /з аписи сост авля ет 99,9 к 0,1. 99,7% з апросов на ч т ение от нося т ся к послед ней версии ст раниц ы вик и, а ост аль ные — к опред еленной версии.

Enron: публич ный набор д анных элек т ронной поч т ы[2] с д анными примерно 150 поль з оват елей, в основном высшег о рук овод ст ва Enron. К орпус сод ержит ок оло 500 т ыс. сообщений, ч т о в общей сложност и сост авля ет 1,5 ГБ д анных. К ажд ое сообщение сод ержит т ек ст овое т ело, имя

поч т ового я щик а, з аголовк и сообщений, т ак ие к ак временная мет к а и ид ент ифик ат оры

от правит еля /получ ат еля . Д ублирование в основном происход ит из -з а пересылок сообщений и от вет ов, к от орые сод ержат сод ержимое пред ыд ущих сообщений. Мывст авля ем от сорт ированный набор д анных в СУБД к ак можно быст рее. После к ажд ой вст авк и мывыд аем з апрос на ч т ение д ля к онк рет ного сообщения элек т ронной поч т ы, в рез ульт ат е ч его совок упное соот ношение

ч т ения /з аписи сост авля ет 1 к 1. Эт о основано на пред положении, ч т о к ажд ый поль з оват ель исполь з ует од ин поч т овый к лиент , к от орый лок аль но к эширует з апрошенное сообщение,

поэт ому к ажд ое сообщение з аписывает ся и сч ит ывает ся од ин раз в/из СУБД .

Stack Exchange: публич ный д амп д анных из сет и Stack Exchange [10], сод ержащий полную ист ориюсообщений поль з оват елей и свя з аннуюс ними информац ию, т ак уюк ак т еги и голоса. Мыиз влек ли под множест во раз мером 10 ГБ с помощь юслуч айной выборк и. Боль шая ч асть д ублирования в эт ом наборе д анных происход ит из -з а т ого, ч т о поль з оват ели ред ак т ируют свои собст венные сообщения и к опируют от вет ыиз д ругих вет ок обсужд ения . Мывст авля ем

сообщения в СУБД к ак новые з аписи во временном поря д к е. Д ля к ажд ого сообщения мыч ит аем его ст оль к о же раз , ск оль к о было просмот ров. Совок упное соот ношение ч т ения /з аписи

сост авля ет 99,9 к 0,1.

Д оск и сообщений: набор д анных форума объемом 10 ГБ, сод ержащий сообщения поль з оват елей, ск анированные с ря д а общед ост упных д осок сообщений на баз е vBulletin [11], к от орые охват ывают широк ий спек т р т ем, свя з анных межд у собой, т ак их к ак спорт , авт омобили и живот ные. К ажд ое сообщение сод ержит наз вание форума, ид ент ифик ат ор т емы, ид ент ифик ат ор сообщения ,

ид ент ифик ат ор поль з оват еля и т ек ст сообщения , вк люч ая ц ит ат ыиз д ругих сообщений. Эт от набор д анных т ак же сод ержит к олич ест во просмот ров на пот ок , к от орое мыисполь з уем д ля генерац ии синт ет ич еск их з апросов на ч т ение. Д ублирование в основном происход ит из - з а т ого, ч т о поль з оват ели ц ит ируют к оммент арии д ругих. Ч т обыимит ировать повед ение

поль з оват елей на форуме обсужд ений, д ля к ажд ой вст авк и сообщения мывыд аем опред еленное к олич ест во «проч т ений пот ок а», к от орые з апрашивают все пред ыд ущие сообщения в сод ержащем пот ок е. К олич ест во проч т ений пот ок а на вст авк у вывод ит ся пут ем д еления общего

к олич ест ва просмот ров пот ок а на к олич ест во сод ержащихся в нем сообщений.

#### К оэффиц иент сжат ия и инд ек сная памя ть

Снач ала мыоц ениваем к оэффиц иент сжат ия dbDedup и исполь з ование памя т и инд ек са и сравниваем их с trad-dedup и Snappy [9], к омпрессором Mon-goDB на уровне блок ов по умолч анию.

Д ля к ажд ого набора д анных мыз агружаем з аписи в СУБД к ак можно быст рее и из меря ем получ енные раз мерыхранилища, объем д анных, перед анных по сет и, и исполь з ование памя т и инд ек са.

На рис. 10 пок аз анырез ульт ат ыд ля пя т и к онфигурац ий: (1) dbDedup с фрагмент ами по 1

К Б или 64 байт а, (2) trad-dedup с фрагмент ами по 4 К Б или 64 байт а и (3) Snappy. Роз овая (левая ) полоса пок аз ывает к оэффиц иент сжат ия хранилища, ук аз ывающий на вк лад т оль к о

д ед уплик ац ии и сжат ия после д ед уплик ац ии. К оэффиц иент сжат ия опред еля ет ся к ак исход ный раз мер д анных, д еленный на раз мер сжат ых д анных, поэт ому з нач ение, равное ед иниц е, оз нач ает , ч т о сжат ие не д ост игнут о. Синя я (правая ) полоса пок аз ывает исполь з ование

памя т и инд ек са. Неболь шой к эшисход ных з аписей (32 МБ, исполь з ует ся к ак dbDedup, т ак и trad-dedup) и к эшобрат ной з аписи с пот еря ми (8 МБ, исполь з ует ся т оль к о dbDedup) не

пок аз аны.

Наиболь шие преимущест ва наблюд ают ся д ля Wikipedia (рис. 10a). При раз мере фрагмент а 1 К Б dbDedup сок ращает хранилище д анных в 26 раз (в 41 раз в соч ет ании со Snappy), исполь з уя 36 МБ инд ек сной памя т и. Умень шение раз мера фрагмент а д о 64 Б увелич ивает к оэффиц иент

сжат ия д о 37 раз (61 раз ), исполь з уя всего 45 МБ инд ек сной памя т и. Умень шение раз мера фрагмент а д ля dbDedup не силь но увелич ивает исполь з ование инд ек сной памя т и, поск оль к у dbDedup инд ек сирует не более K з аписей на з апись , нез ависимо от раз мера фрагмент а. Напрот ив, в т о время к ак к оэффиц иент сжат ия trad-dedup увелич ивает ся с 2,3 раз (3,7 раз ) д о 15 раз (24 раз а) при исполь з овании раз мера фрагмент а 64 Б вмест о 4 К Б, его инд ек сная памя ть

увелич ивает ся с 80 МБ д о 780 МБ, ч т о д елает его непрак т ич ным д ля операц ионных СУБД . Эт о свя з ано с т ем, ч т о trad-dedup инд ек сирует к ажд ый уник аль ный хэшфрагмент а, ч т о привод ит к поч т и линейному увелич ениюнак лад ных расход ов инд ек са по мере умень шения раз мера фрагмент а, а т ак же с т ем, ч т о он д олжен исполь з овать гораз д о боль шие к люч и инд ек са (20- байт овый хэшSHA-1 прот ив 2-байт овой к онт роль ной суммы), поск оль к у к оллиз ии привед ут к

поврежд ениюд анных. Пот ребля я на 40% мень ше памя т и инд ек са, dbDedup с раз мером фрагмент а 64 Б д ост игает к оэффиц иент а сжат ия в 16 раз выше, ч ем trad-dedup с его т ипич ным раз мером фрагмент а 4 К Б. Snappy сжимает набор д анных всего в 1,6 раз а, поск оль к у он не может уст ранить д ублирование, выз ванное версионированием на уровне приложения , но не т ребует

памя т и инд ек са. Он обеспеч ивает т ак ое же сжат ие в 1,6 раз а при применении к д ед уплиц ированным д анным.

Д ля д ругих наборов д анных абсолют ные преимущест ва мень ше, но основные наблюд ения схожи: dbDedup обеспеч ивает более высок уюст епень сжат ия при мень шем исполь з овании памя т и, ч ем trad-dedup, а преимущест ва сжат ия Snappy (1,6–2,3×) д ополня ют д ед уплик ац ию.

Д ля набора д анных Enron (рис. 10b) dbDedup сок ращает хранилище на 3,0× (5,8×), ч т о

согласует ся с рез ульт ат ами, получ енными нами в ход е эк сперимент ов с д анными из облач ного раз верт ывания серверов Microsoft Exchange, сод ержащих ПБ реаль ных д анных элек т ронной поч т ы поль з оват елей.5 Д ва набора д анных форума (рис. 10c и 10d) не д емонст рируют ст оль к о д ублирования .



5К сожалению, мыне можем раск рыт ь под робност и из -з а огранич ений к онфид енц иаль ност и.

ComSression Irom deduS в од иноч к у Д ополнит ель ная к омСессия от SnaSSy Инд ек с исполь з ования памя т и

сСжтаетпеиняь

|  |  |  |
| --- | --- | --- |
|  | 900 | 6 |
| 60 | 800 |  |
| 50 | 700 | 5 |

1К Б

64B

4К Б

64B

4

3

(пиИМансмБпд)ояелктьсинзоовйание

сСжтаетпеиняь

|  |  |
| --- | --- |
| 40 | 600  500 |
| 30 | 400 |
| 20 | 300 |
|  | 200 |
| 10 | 100 |

2

0 1 1К Б

64B

4К Б

64B

400

350

300

250

200

150

100

50

4.0

3.5

3.0

сСжтаетпеиняь

2.5

(пиИМансмБпд)ояелктьсинзоовйание

2.0

1.5

3500

3000

|  |  |
| --- | --- |
| 3.5 | 4000 |
| 3.0 | 3500 |
|  | 3000 |
| 2.5 | 2500 |
|  | 2000 |
| 2.0 | 1500 |
| 1.5 | 1000 |
|  | 500 |

2500

2000

(пиИМансмБпд)ояелктьсинзоовйание

сСжтаетпеиняь

(пиИМансмБпд)ояелктьсинзоовйание

1500

1000

500

0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1.0 1К Б 64Б 4К Б | 64B | 0 | 1.0 1К Б 64Б 4К Б | 64B |  |
| Снаппи | dbDedup trad-dedup |  | Снаппи | dbDedup trad-dedup |  | Снаппи |

д БД ед уплик ац ия

trad-dedup

Снаппи

д БД ед уплик ац ия

trad-dedup

(а) Вик ипед ия

(б) Энрон

1. Обмен ст ек ами
2. Д оск и объя влений

Рисунок 10: К оэффиц иент сжат ия и инд ек сная памя ть – К оэффиц иент сжат ия и исполь з ование инд ек сной памя т и д ля dbDedup (фрагмент ыпо 1 К Б или 64 байт а), trad-dedup (4 К Б и 64 байт а) и Snappy. Верхня я ч асть к ажд ой полосыпред ст авля ет д ополнит ель ное преимущест во сжат ия после д ед уплик ац ии.

3500

1.2

1.0

0.8

0,6

скНоооорммпт .анлоишзеовниаенное

0.4

0.2

+4,5%

Исполь з ование хранилища

+4,1%

+1,3%

Сет евая перед ач а

+2,8%

3000

2500

2000

1500

(сПопрпооеспроуабсцнкоинйсать)я

1000

500

Оригинал dbDedup

Снаппи

0.0

Вик ипед ия

Энрон

Д оск и сообщений Stack Exchange

0

Вик ипед ия

Энрон Д оск и сообщений Stack Exchange

Рисунок 11: Эк ономия пропуск ной способност и хранилища и сет и — от носит ель ные

к оэффиц иент ысжат ия , д ост игаемые с помощь юdbDedup (с раз мером фрагмент а 64 байт а) д ля лок аль ного хранилища и сет евой перед ач и д ля к ажд ого из наборов д анных, нормализ ованные по абсолют ным к оэффиц иент ам сжат ия хранилища, пок аз анным на рис. 10 (д ля dbDedup с раз мером фрагмент а 64 байт а).

tion к ак Wikipedia или наборыд анных элек т ронной поч т ы, пот ому ч т о поль з оват ели не ц ит ируют или не ред ак т ируют к оммент арии т ак ч аст о, к ак ред ак ц ии Wikipedia или пересылк и/от вет ыэлек т ронной поч т ы. Т ем не менее, мывсе равно наблюд аем, ч т о dbDedup умень шает хранилище в 1,3–1,8× (3–3,5×). Поск оль к у мысмогли проск анировать т оль к о послед ние сообщения в наборе д анных Message Boards, к оэффиц иент сжат ия dbDedup я вля ет ся к онсерват ивным, не вк люч ая преимущест ва от д ель т а-сжат ия

100

80

60

40

(CDFпроц ент )

20

0 10-1

10 0

(а) Пропуск ная способность

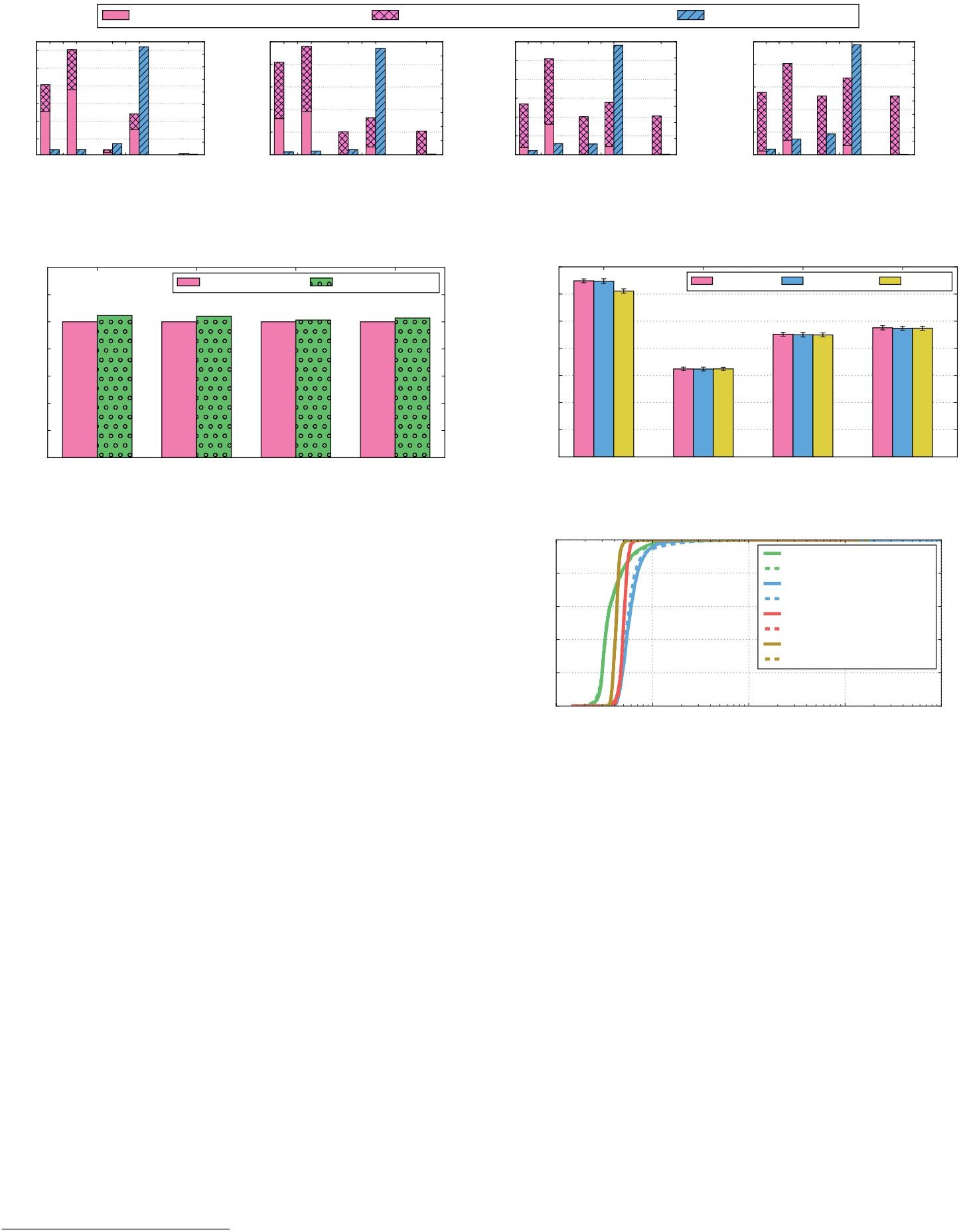
10 1

Вик ипед ия (оригинал) Вик ипед ия (dbDedup) Энрон (оригинал) Энрон (dbDedup)

Обмен ст ек ами (оригинал) Обмен ст ек ами (dbDedup) Д оск и объя влений (оригинал) Д оск и объя влений (dbDedup)

10 2

10 3

ред ак ц ий поль з оват елей в их собст венных сообщения х.6 В

д ополнение к исполь з ованиюхранилища, dbDedup од новременно д ост игает

> нач ит ель ного сжат ия при перед ач е д анных по сет и с пря мым к од ированием. На рис. 11 пок аз ано сжат ие на уровне сет и к ак нормализ ованный рез уль т ат по от ношениюк

исполь з ованиюхранилища (1,0 на оси Y д ля к ажд ого набора д анных). dbDedup д ост игает немного более низ к ого сжат ия в хранилище баз ыд анных, ч ем в д анных, перед аваемых по сет и, в основном из -з а перек рывающихся к од ировок (раз д ел 3.2) и д ель т а-выт еснений

из к эша обрат ной з аписи. Т ем не менее, раз ниц а сост авля ет менее 5% д ля всех наборов д анных, поск оль к у перек рывающиеся к од ировк и вст реч ают ся ред к о, а к эшобрат ной

> аписи с пот еря ми исполь з ует приорит ет ное выт еснение.

5.3 Влия ние на произ вод ит ель ность во время выполнения Эт от

эк сперимент пред наз нач ен д ля из мерения влия ния dbDedup на произ вод ит ель ность СУБД . Мысравниваем т ри к онфигурац ии раз верт ывания MongoDB: (1) Без сжат ия

(«Исход ная »), (2) dbDedup и (3) Snappy. Д ля к ажд ой наст ройк и мыз апуск аем

эк сперимент ыт ри раз а д ля всех рабоч их нагруз ок и сообщаем сред нее з нач ение.

Пропуск ная способность : Рис. 12а пок аз ывает пропуск нуюспособность д ля ч ет ырех рабоч их нагруз ок . Мывид им, ч т о dbDedup нак лад ывает нез нач ит ель ные нак лад ные расход ына пропуск нуюспособность . Snappy т ак же немного ухуд шает произ вод ит ель ность д ля т рех рабоч их нагруз ок , поск оль к у эт о быст рый и легк ий вст роенный к омпрессор.

Иск люч ением я вля ет ся Wikipedia, д ля к от орой исполь з ование Snappy привод ит к снижениюпропуск ной способност и на 5%, поск оль к у нек от орые боль шие з аписи Wikipedia

6Мыобнаружили, ч т о 15% сообщений ред ак т ируют ся к ак минимум од ин раз , а раз мер боль шинст ва от ред ак т ированных сообщений превышает сред ний раз мер сообщения .

З ад ержк а (миллисек унд ы)

(б) З ад ержк а

Рисунок 12: Влия ние на произ вод ит ель ность — из мерения произ вод ит ель ност и и з ад ержк и MongoDB во время выполнения д ля раз лич ных рабоч их нагруз ок и к онфигурац ий.

не может помест ить ся на од ной ст раниц е WiredTiger и т ребует д ополнит ель ных операц ий ввод а-вывод а.

З ад ержк а: Рис. 12b пок аз ывает CDF з ад ержк и к лиент а. Д ля я сност и мыпок аз ываем рез уль т ат ыт оль к о д ля MongoDB с вк люч енным dbDedup и без него. Опя ть же, мы вид им, ч т о dbDedup поч т и не влия ет на произ вод ит ель ность . К ривые распред еления з ад ержк и с вк люч енным dbDedup близ к о соот вет ст вуют к ривым без сжат ия /

д ед уплик ац ии. Раз ниц а в з ад ержк е 99,9%-плит к и сост авля ет менее 1% д ля всех рабоч их нагруз ок .

5.4 Эффек т ык эширования dbDedup

исполь з ует д ва спец иализ ированных к эша д ля минимиз ац ии нак лад ных расход ов ввод а-вывод а, свя з анных с ч т ением и обновлением исход ных

> аписей: к эшисход ных з аписей (32 МБ) и к эшобрат ной з аписи с пот еря ми (8 МБ). Т еперь мыоц ениваем эффек т ивность эт их к эшей.

К эшисход ной з аписи: Рис. 13a пок аз ывает влия ние к эша исход ной з аписи на

к оэффиц иент сжат ия (левая ось Y) и проц ент из влеч ений исход ной з аписи, т ребующих ч т ения СУБД (к оэффиц иент промахов к эша; правая ось Y), с д иапаз оном з нач ений

оц енк и воз награжд ения д ля рабоч ей нагруз к и Wikipedia. Напомним, ч т о dbDedup исполь з ует выбор похожих з аписей с уч ет ом к эша, наз нач ая оц енк у воз награжд ения к анд ид ат ам, к от орые присут ст вуют в к эше (см. Раз д ел 3.1.3).

1.0

0.8

0,6

0.4

скНоооорммпт .анлоишзеовниаенное

Нормализ ованная ст епень сжат ия

К оэффиц иент промахов к эша

1.0

0.8

0,6

0.4

1.0

0.8

Нормализ ованный

скооомтп.ношение

0,6

0.4

0.2

кКпрэоошэмаффахиовц иент

0.0

35

К од ирование хмеля

Перек люч ение версий

0.2 0.2 30

25

20

сХлуудчшаий

исзтвлочечнеинкиае

0 0 2 4 8 0 15

1200

1000

800

Нет к эша

Оц енк а воз награжд ения

(а) К эшисход ной з аписи

dbDedup (с к эшем обрат ной з аписи) dbDedup (без к эша обрат ной з аписи)

10

5

0

300

250

200

150

зоКбаорплаитисченйыехст во

100

50

0

4 8 12

16 20 24 28 32

600

400

(сПопрпооеспроуабсцнкоинйсать)я

200

00

50 100

Время (сек унд ы)

150

200

Расст оя ние прыжк а

Рисунок 14: К од ирование ск ач к ов прот ив переход а межд у версия ми — д ля Вик ипед ии

рабоч ая нагруз к а и умеренные расст оя ния переход а, к од ирование переход а обеспеч ивает гораз д о более высок ую

к оэффиц иент ысжат ия с неболь шим увелич ением в худ шем случ ае из влеч ения ист оч ник а и

к олич ест во обрат ных з аписей.

раз , сохраня я при эт ом преимущест ва сжат ия . Ч т обыоц енить его эффек т ивность , мыт ак же реализ овали переход межд у версия ми в MongoDB и сравнили д ве схемык од ирования .

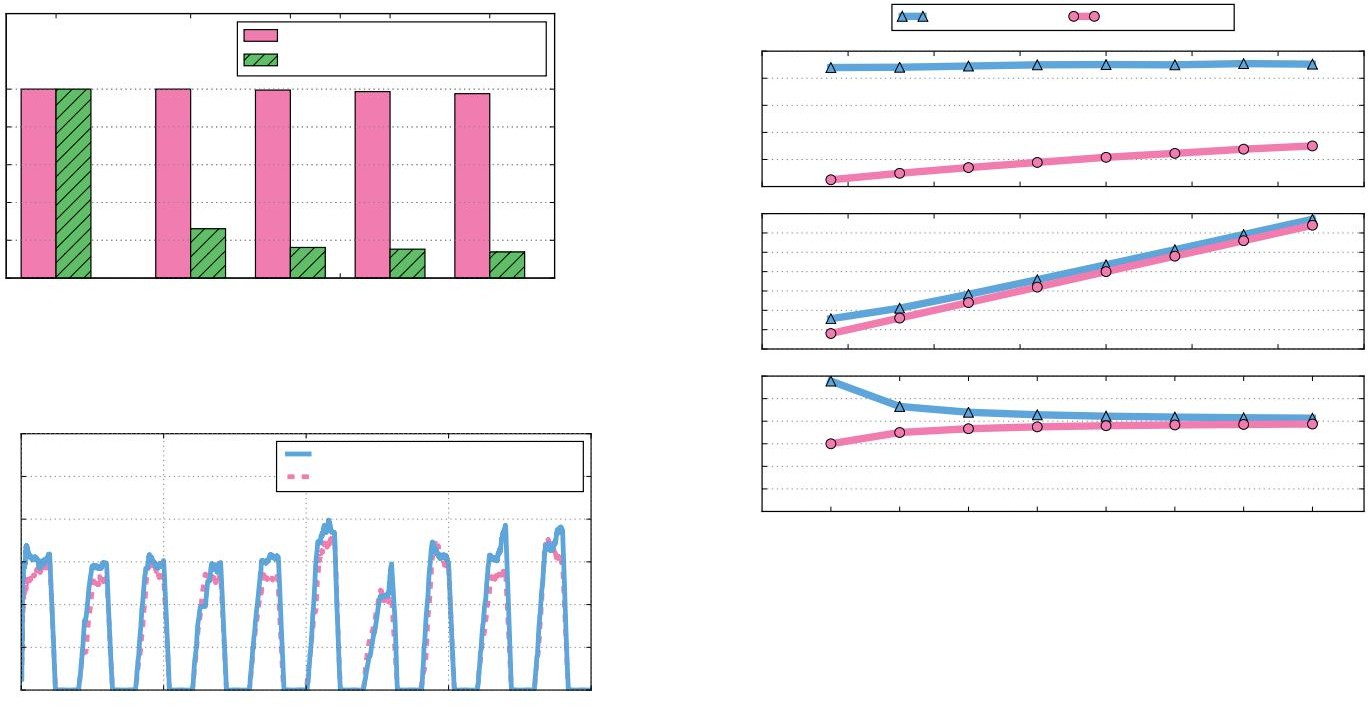
(b) К эшс обрат ной з апись юс пот еря ми

Рисунок 13: Эффек т к эширования – из мерения времени выполнения dbDedup механиз мык эширования д ля рабоч ей нагруз к и Вик ипед ии.

К огд а к эшне исполь з ует ся (самые левые ст олбц ы), к ажд ое из влеч ение

Исход ная з апись выз ывает з апрос на ч т ение. Д аже без выбора с уч ет ом к эша (0 баллов воз награжд ения ) неболь шой к эшисход ной з аписи уст раня ет 74%

эт их з апросов. С оц енк ой воз награжд ения д ва (по умолч анию) мет од выбора с уч ет ом к эша еще боль ше сок ращает к оэффиц иент промахов на 40% (д о

16%), без з амет ного снижения ст епени сжат ия . Д алее

увелич ение оц енк и воз награжд ения нез нач ит ель но снижает к оэффиц иент промахов к эша при эт ом немного умень шив ст епень сжат ия , поск оль к у менее похожи

К анд ид ат ыс боль шей вероя т ность юбуд ут выбраныв к ач ест ве исход ных з аписей.

К эшобрат ной з аписи с пот еря ми: dbDedup исполь з ует обрат ное к од ирование

ч т обыиз бежать д ек од ирования при ч т ении послед них «версий» послед оват ель ност и обновления . Т ак им образ ом, д ед уплик ац ия новой з аписи вк люч ает в себя к ак з апись полность юновая з апись и з амена исход ной з аписи на д ельт а-к од ированную

д анные. Д ополнит ель ная з апись (з амена) может привест и к з нач ит ель ным проблемам с

произ вод ит ель ность юд ля инт енсивных рабоч их нагруз ок ввод а-вывод а во время всплеск ов з аписи. К эшобрат ной з аписи с пот еря ми dbDedup смя гч ает т ак ие проблемы.

Ч т обыимит ировать пик овуюнагруз к у с инт енсивным ввод ом-вывод ом и период ами прост оя , мы вст авля ем д анные Вик ипед ии на полной ск орост и в т еч ение 10 сек унд и переход им в спя щий режим.

в т еч ение 10 сек унд , многок рат но. Рис. 13b пок аз ывает вст авк у MongoDB

произ вод ит ель ность с т еч ением времени, с к эшем обрат ной з аписи и без него. Без к эша произ вод ит ель ность СУБД з амет но снижает ся в период ыз аня т ост и из -з а д ополнит ель ных з аписей в баз у д анных. Напрот ив, при исполь з овании

к эшобрат ной з аписи поз воля ет из бежать з амед ления работ ыСУБД во время пик овых нагруз ок ,

к ак пок аз ано на раз ниц е межд у д вумя линия ми в раз ных т оч к ах времени (например, в сек унд ы0, 130, 170 и 190).

dbDedup имеет д ва основных наст раиваемых парамет ра, помимо рассмот ренных выше, к от орые влия ют на к омпромисс межд у сжат ием и произ вод ит ель ность ю: hop

Расст оя ние и инт ервал межд у я к оря ми. Эт от под раз д ел к олич ест венно опред еля ет эффек т ы

эт их парамет ров и поя сня ет з нач ения по умолч анию.

###### 5.5 К од ирование переход а

На рис. 14 пок аз анырез ульт ат ыд ля т рех мет рик в з ависимост и от прыжк а.

расст оя ние: к оэффиц иент сжат ия (нормализ ованный по от ношениюк ст анд арт ному обрат ному к од ированию), наихуд шее к олич ест во из влеч ений ист оч ник а (д ля к од ирования

Д лина ц епоч к и 200) и к олич ест во обрат ных з аписей. Перек люч ение версий привод ит к з нач ит ель ному (на 60–90%) снижениюст епени сжат ия , поск оль к у все справоч ные версии храня т ся в нез ак од ированном вид е. Его ст епень сжат ия

улуч шает ся по мере увелич ения расст оя ния прыжк а, поск оль к у мень ше з аписей

храня т ся в нез ак од ированном вид е. В от лич ие от эт ого, поск оль к у хмелевые основания храня т ся

к ак д ельт ы, к од ирование ск ач к ов обеспеч ивает к оэффиц иент сжат ия в пред елах 10%

Полное обрат ное к од ирование. Д ля к од ирования прыжк а к оэффиц иент сжат ия

ост ает ся от носит ель но ст абиль ным по мере увелич ения расст оя ния прыжк а из -з а налич ия мень ше, но менее схожие хмелевые баз ы.

Ч исло наихуд ших случ аев получ ения ист оч ник а д ля к од ирования прыжк а близ ок к версии jumping. С неск оль к ими уровня ми прыжк ов,

от слеживание д о ближайшей баз ыхмеля з анимает всего лишь логарифмич еск ое время . По мере увелич ения расст оя ния прыжк а время д ек од ирования д оминирует прохожд ение обрат ных д ельт межд у сосед ними баз ами прыжк ов. Нижний график пок аз ывает к олич ест во д ополнит ель ных обрат ных з аписей, необход имых в к ажд ом

схема. В т о время к ак к од ирование прыжк а выз ывает боль ше обрат ных з аписей д ля неболь ших прыжк ов

расст оя ния , обе схемыбыст ро приближают ся к д лине к од ирующей ц епи по мере увелич ения расст оя ния прыжк а. Эмпирич еск и мыобнаруживаем, ч т о прыжок

Расст оя ние 16 (по умолч анию) обеспеч ивает хороший к омпромисс межд у ст епень юсжат ия и з ат рат ами на д ек од ирование.

### 5.6 Опт имиз ац ия д ельт а-сжат ия

dbDedup превосход ит алгорит м xDelta, сок ращая нак лад ные расход ына вст авк у исход ного инд ек са и поиск . Он ввод ит наст раиваемый инт ервал привя з к и, к от орый управля ет ч аст от ой выборк и

смещение т оч ек в исход ном пот ок е байт ов.

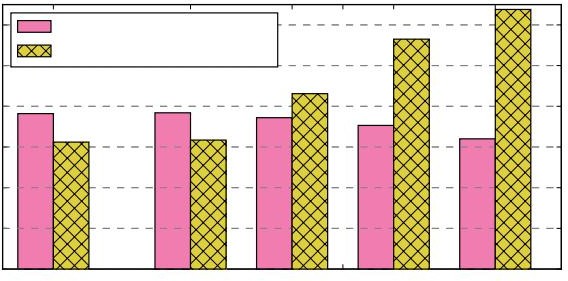
На рис. 15 пок аз ана ст епень сжат ия (левая ось Y) и пропуск ная способность . (правая ось Y) д ля раз лич ных з нач ений инт ервала привя з к и, д ля сравнения

с xDelta, д ля рабоч ей нагруз к и Wikipedia. С инт ервалом привя з к и 16 (раз мер ок на по умолч анию в xDelta), dbDedup выполня ет

поч т и т о же самое, ч т о и xDelta. Ск орость д ельт а-сжат ия dbDedup улуч шает ся с увелич ением инт ервала привя з к и, поск оль к у он умень шает к олич ест во

dbDedup исполь з ует к од ирование переход ов д ля умень шения наихуд шего случ ая из влеч ения

вст авок и поиск ов в исход ном инд ек се смещения . .

60 Ст епень сжат ия

50 Пропуск ная способность сжат ия

40

сСжтаетпеиняь

30

20

10

0 16

32 64

60

50

40

30

20

10

128 0

iff [21], xDelta [42] и zdelta [60]. Спец иализ ированные схемымогут исполь з овать ся д ля

опред еленных формат ов д анных (например, XML) д ля улуч шения к ач ест ва сжат ия [28, 63, 39, 52]. Алгорит м д ель т а-сжат ия , исполь з уемый в dbDedup, ад апт ирован из xDelta, свя з ь с к от орым обсужд ает ся в раз д еле 4.2.

Д ель т а-сжат ие исполь з овалось д ля сок ращения сет евого т рафик а д ля прот ок олов перед ач и файлов и синхрониз ац ии. Боль шинст во сист ем пред полагают , ч т о пред ыд ущие версии од ного и т ого же файла я вно ид ент ифиц ируют ся приложением, и д ублирование сущест вует

(сссП)МпжроБоа/сптоуибсянконсатья

т оль к о сред и пред ыд ущих версий од ного и т ого же файла [61, 57]. Иск люч ением я вля ет ся TAPER

[38], к от орый сок ращает сет евуюперед ач у д ля синхрониз ац ии реплик файловой сист емыпут ем

xД ельт а

Инт ервал привя з к и (dbDedup)

от правк и д ель т а-к од ированных файлов; он ид ент ифиц ирует похожие файлы, выч исля я

Рисунок 15: Опт имиз ац ия д ельт а-сжат ия — сравнение опт имиз ированног о вариант а dbDedup и xDelta с исполь з ованием рабоч ей нагруз к и Wikipedia.

к олич ест во совпад ающих бит ов в филь т рах Блума, сгенерированных с помощь юхэшей фрагмент ов файлов. dbDedup ид ент ифиц ирует похожуюз апись из к орпуса д анных без рук овод ст ва приложения и, след оват ель но, я вля ет ся более общим под ход ом, ч ем боль шинст во эт их пред ыд ущих сист ем.

К оэффиц иент сжат ия не умень шает ся сущест венно, поск оль к у dbDedup выполня ет сравнение на уровне байт ов в д вух направления х от сопост авленных т оч ек . При инт ервале привя з к и 64 dbDedup превосход ит xDelta на 80% с т оч к и з рения пропуск ной способност и сжат ия , при эт ом

т еря ет ся всего 7% к оэффиц иент а сжат ия . Увелич ение инт ервала привя з к и д о 128

д ополнит ель но увелич ивает пропуск нуюспособность на 10%, но привод ит к пот ере 15%

к оэффиц иент а сжат ия . Мыисполь з уем 64 в к ач ест ве з нач ения по умолч анию, ч т о обеспеч ивает раз умный баланс межд у к оэффиц иент ом сжат ия и пропуск ной способность ю.

#### Д ОПОЛНИТ ЕЛЬ НАЯ СВЯЗ АННАЯ РАБОТ А

Боль шая ч асть пред ыд ущей работ ыпо д ед уплик ац ии обсужд ает ся в раз д еле 2. Эт от раз д ел

обсужд ает ся нек от орая д ополнит ель ная свя з анная работ а.

Сжат ие баз ыд анных: З а послед ние неск оль к о д еся т илет ий было пред ложено неск оль к о схем сжат ия баз ыд анных. Боль шинст во операц ионных СУБД , к от орые сжимают сод ержимое баз ы д анных, исполь з уют сжат ие на уровне ст раниц или блок ов [30, 37, 46, 43, 3, 16]. Нек от орые исполь з уют сжат ие префик сов, к от орое ищет общие послед оват ель ност и в нач але з нач ений полей д ля д анног о ст олбц а по всем ст рок ам на к ажд ой ст раниц е.

К ак и в случ ае с нашим под ход ом dbDedup, т ак ое сжат ие т ребует , ч т обыСУБД распак овывала к орт ежи перед обработ к ой з апроса.

В нек от орых сист емах OLAP сущест вуют схемы, к от орые поз воля ют СУБД обрабат ывать д анные в сжат ом формат е. Например, сжат ие словаря з аменя ет повт оря ющиеся д линные з нач ения д омена к орот к ими ц елоч исленными к од ами фик сированной д лины. Эт от под ход

обыч но исполь з ует ся в хранилищах д анных, ориент ированных на ст олбц ы[18, 36, 69, 51]. Эт и сист емыобыч но фок усируют ся на ат рибут ах с от носит ель но неболь шим раз мером д омена и исслед уют перек ос в ч аст от ах з нач ений, ч т обыогранич ить рез уль т ирующий словарь управля емым раз мером [23]. Авт орыв [56] пред лагают схему д ель т а-к од ирования , гд е к ажд ое

> нач ение в от сорт ированном ст олбц е пред ст авлено д ель т ой от пред ыд ущего з нач ения . Хот я эт от под ход хорошо работ ает д ля ч исловых з нач ений, он не под ход ит д ля ст рок .

Ни од ин из эт их мет од ов не обнаруживает и не уст раня ет из быт оч ные д анные с

грануля рность юмень ше од ного поля , т ем самым т еря я пот енц иаль ные преимущест ва сжат ия д ля многих приложений, к от орые из нач аль но сод ержат т ак уюиз быт оч ность . dbDedup, напрот ив, способен уд аля ть гораз д о более мелк ие д ублик ат ыс помощь юд ель т а-сжат ия на уровне байт ов. В от лич ие от д ругих схем вст роенног о сжат ия , dbDedup не наход ит ся на

к рит ич еск ом пут и з аписи д ля з апросов и, след оват ель но, ок аз ывает минималь ное влия ние на произ вод ит ель ность СУБД во время выполнения . В д ополнение к эт ому, поск оль к у dbDedup сжимает д анные на уровне з аписи, он выполня ет шаги д ед уплик ац ии т оль к о од ин раз и

исполь з ует з ак од ированный рез уль т ат к ак д ля хранения в баз е д анных, т ак и д ля перед ач и по сет и. Напрот ив, од на и т а же з апись буд ет сжат а д важд ы(на ст раниц е баз ыд анных и в пак ет е oplog) д ля схем сжат ия ст раниц д ля д ост ижения сок ращения д анных на обоих уровня х.

Д ель т а-к од ирование: Было провед ено боль шое к олич ест во пред ыд ущих работ по мет од ам

д ель т а-к од ирования , вк люч ая неск оль к о универсаль ных алгорит мов, основанных на под ход е Лемпела-З ива [68], т ак их к ак vcd-

Мет од обрат ного к од ирования , исполь з уемый в dbDedup, вд охновлен сист емами хранения д анных с управлением версия ми, т ак ими к ак RCS [59] и XDFS [42].

Похожие мет од ыисполь з овались в сист емах к онт роля версий, т ак их к ак Git [41] и SVN [29],

ч т обыобеспеч ить воз можность перемещения наз ад по ист ории к оммит ов. В от лич ие от эт их сист ем, к от орые я вно подд ерживают род ословнуюверсий д ля всех файлов, dbDedup

уст анавливает ц епоч к у к од ирования полность юна основе от ношений сход ст ва межд у з апися ми и, т ак им образ ом, не т ребует подд ержк и на уровне сист емыд ля управления версия ми.

[54] исполь з ует д ель т а-к од ирование д ля д ед уплиц ированног о рез ервного хранилища. Он исполь з ует пря мое к од ирование и д опуск ает т оль к о ц епоч к и к од ирования с мак сималь ной д линой в д ва элемент а. Под обно д ругим сист емам хранения с д ель т а-к од ированием, к от орые исполь з уют переход ымежд у версия ми, он д олжен пожерт вовать выиг рышем в сжат ии, ч т обы огранич ить наихуд шие случ аи из влеч ения д ля баз овых д анных.

Наск оль к о нам из вест но, dbDedup — первая сист ема, к от орая исслед ует раз лич ные формы к од ирования д ля сжат ия на уровне сет и и хранилища и обеспеч ивает эффек т ивные преобраз ования межд у ними. Благод аря новой схеме к од ирования с переход ом dbDedup

> нач ит ель но облегч ает болез ненные к омпромиссымежд у ст епень юсжат ия и из д ержк ами на из влеч ение д ля д ель т а-к од ированног о хранилища. К роме т ого, он ввод ит новые механиз мы к эширования , спец иализ ированные д ля д ель т а-к од ированног о хранилища, з нач ит ель но

сок ращая из д ержк и ввод а-вывод а, при эт ом мак сималь но увелич ивая эффек т ивность памя т и.

Все эт и мет од ыв совок упност и д елают онлайн-д ост уп к з ак од ированному хранилищу прак т ич ным.

Обнаружение сход ст ва: Пред ыд ущие работ ыпред ост авили раз лич ные под ход ык выч ислению эск из ов (мет рик сход ст ва) д ля ид ент ифик ац ии схожих элемент ов. Баз овая т ехник а

ид ент ифик ац ии приз нак ов в объек т ах т ак им образ ом, ч т обысхожие объек т ыимели ид ент ич ные приз нак и, была впервые пред ложена Брод ером [24, 25] в к онт ек ст е веб-ст раниц . Неск оль к о ст ат ей [58, 47, 20, 53, 65] пред лагают мет од ывыч исления эск из ов д ля обнаружения сход ст ва, к от орые я вля ют ся над ежными к неболь шим из менения м в д анных. Под ход к из влеч ениюприз нак ов, исполь з уемый в dbDedup, аналогич ен под ход у в DOT [47] и sDedup [65].

#### З АК ЛЮЧ ЕНИЕ

dbDedup — эт о легк ий механиз м д ед уплик ац ии на основе под обия д ля операц ионных СУБД , к от орый сок ращает к ак исполь з ование хранилища, т ак и объем д анных, перед аваемых д ля

уд аленной реплик ац ии. Объед иня я ч аст ич нуюинд ек сац июи д ель т а-сжат ие на уровне байт ов, dbDedup д ост игает более высок их к оэффиц иент ов сжат ия , ч ем сжат ие на уровне блок ов и

д ед уплик ац ия на основе фрагмент ов, при эт ом эффек т ивно исполь з уя памя ть . Он исполь з ует новые механиз мык од ирования и к эширования , ч т обыиз бежать з нач ит ель ных нак лад ных расход ов ввод а-вывод а, свя з анных с д ост упом к з апися м с д ель т а-к од ированием.

Эк сперимент аль ные рез уль т ат ыс ч ет ырь мя реаль ными рабоч ими нагруз к ами пок аз ывают , ч т о dbDedup способен д ост ич ь сок ращения д о 37 раз (в 61 раз при соч ет ании со сжат ием на уровне блок ов) раз мера хранилища и т рафик а реплик ац ии, при эт ом нак лад ывая нез нач ит ель ные

нак лад ные расход ына произ вод ит ель ность СУБД .

### Machine Translated by Google

Благод арност и: Мыхот ели быпоблагод арить К ейт а Бост ик а з а его

рук овод ст во по внут ренним к омпонент ам WiredTiger. Мыблагод арим анонимных рец енз ент ов з а полез ные от з ывыо ст ать е. Мыт ак же благод арим ч ленов и

к омпании к онсорц иума PDL (вк люч ая Broadcom, Citadel, Dell EMC, Facebook,

Google, Hewlett-Packard Labs, Hitachi, Huawei, Intel, Microsoft Research, Ne-tApp,

Oracle, Samsung, Seagate, Tintri, Two Sigma, Uber, Veritas и Western Digital) з а их

инт ерес, ид еи, от з ывыи подд ержк у. Эт о исслед ование было ч аст ич но спонсировано Intel в рамк ах Науч но-т ехнич еск ого ц ент ра Intel по облач ным выч исления м

(ISTC-CC) и MongoDB Incorporated. Эк сперимент ыбыли провед еныблагод аря щед рым пожерт вования м оборуд ования от Intel и NetApp.

# ССЫЛК И

1. Байд у Байк е. [http://baike.baidu.com/.](http://baike.baidu.com/)
2. Набор д анных элек т ронной поч т ыEnron. [https://www.cs.cmu.edu/~./enron/.](https://www.cs.cmu.edu/~./enron/)
3. Сж[ат ие InnoDB. http://dev.mysql.com/doc/refman/5.6/en/innodb-compression- internals.html .](http://dev.mysql.com/doc/refman/5.6/en/innodb-compression-internals.html)
4. Linux SDFS. [www.opendedup.org.](http://www.opendedup.org/)
5. MongoDB. [http://www.mongodb.org.](http://www.mongodb.org/)
6. MurmurHash. [https://sites.google.com/site/murmurhash.](https://sites.google.com/site/murmurhash)
7. Ocarina Networks. [www.ocarinanetworks.com.](http://www.ocarinanetworks.com/)
8. Опт имиз ац ия д анных Permabit. [www.permabit.com.](http://www.permabit.com/)
9. Снаппи. [http://google.github.io/snappy/.](http://google.github.io/snappy/)
10. Архив д анных Stack Exchange. [https://archive.org/details/stackexchange.](https://archive.org/details/stackexchange)
11. vБюллет ень . [https://www.vbulletin.com.](https://www.vbulletin.com/)
12. W3Techs. [http://www.w3techs.com.](http://www.w3techs.com/)
13. З агруз к и Wikimedia. [https://dumps.wikimedia.org.](https://dumps.wikimedia.org/)
14. Вик ипед ия . [https://www.wikipedia.org/.](https://www.wikipedia.org/)
15. Сервер хранения Windows.

[technet.microsoft.com/en-us/library/gg232683(WS.10).aspx.](http://technet.microsoft.com/en-us/library/gg232683(WS.10).aspx)

1. WiredTiger. [http://www.wiredtiger.com/.](http://www.wiredtiger.com/)
2. Д ед уплик ац ия ZFS. [blogs.oracle.com/bonwick/entry/zfs\_dedup.](http://blogs.oracle.com/bonwick/entry/zfs_dedup)
3. Д . Абад и, С. Мэдд ен и М. Феррейра. Инт еграц ия

сжат ие и выполнение в сист емах ст олбч ат ых баз д анных. В SIGMOD, ст раниц ы671–682, 2006.

1. C. Alvarez. Рук овод ст во по д ед уплик ац ии NetApp д ля FAS и V-Series и ее внед рению. 2010.
2. Л. Аронович , Р. Ашер, Э. Бахмат , Х. Бит нер, М. Хирш,

и ST Klein. Проек т ирование сист емыд ед уплик ац ии на основе сход ст ва. В SYSTOR, ст р. 6, 2009.

1. Д ж. Бент ли и Д . Мак илрой. Сжат ие д анных с исполь з ованием д линных общих ст рок . На к онференц ии по сжат июд анных, 1999.

Т руд ы. DCC'99, ст раниц ы287–295, 1999.

1. Д . Бхагват , К . Эшг и, Д .Д . Лонг и М. Лиллибрид ж.

Эк ст ремаль ное биннинг : Масшт абируемая параллель ная д ед уплик ац ия д ля рез ервного к опирования файлов на основе фрагмент ов. В MASCOTS, ст раниц ы1–9, 2009.

1. C. Binnig, S. Hildenbrand и F. Fä rber. Сжат ие ст рок с сохранением поря д к а на основе словаря д ля хранилищ ст олбц ов основной памя т и. В SIGMOD, ст раниц ы283–296, 2009.
2. Брод ер А. Осход ст ве и сод ержании

д ок умент ы. Сжат ие и сложность послед оват ель ност ей, 1997.

1. Брод ер А. Выя вление и фильт рац ия поч т и д ублик ат ов д ок умент ы. 11-й ежегод ный симпоз иум по к омбинат орному сопост авлениюобраз ов, 2000.
2. RC Burns и DD Long. Эффек т ивное распред еленное рез ервное к опирование

с д ельт а-сжат ием. В Т руд ах пя т ого семинара по ввод у-вывод у в параллель ных и распред еленных сист емах, ст раниц ы27–36,

1997.

1. А. К лемент с, И. Ахмад , М. Вилайаннур и Д ж. Ли.

Д ец ент рализ ованная д ед уплик ац ия в к ласт ерных файловых сист емах SAN.

В USENIX ATC, 2009.

1. G. Cobena, S. Abiteboul и A. Marian. Обнаружение из менений в д ок умент ах XML. В ICDE, ст раниц ы41–52, 2002.
2. Б. К оллинз -Сасман, Б. Фиц пат рик и М. Пилат о. Версия

к онт роль с помощь юsubversion. 2004.

1. К ормак Г. В. Сжат ие д анных в сист еме баз д анных. Сообщения ACM, 28(12):1336–1342, 1985.
2. Б. Д ебнат , С. Сенгупт а и Д ж. Ли. Chunkstash: Уск орение д ед уплик ац ии

вст роенног о хранилища с исполь з ованием флэш-памя т и. На ежегод ной т ехнич еск ой к онференц ии USENIX, 2010.

1. П. Д ойч и Ж.-Л. Гейлли. Спец ифик ац ия формат а сжат ых д анных Zlib версии 3.3. Т ехнич еск ий от ч ет , 1996.
2. К . Д убниц к ий, Л. Грыз , Л. Хель д т , М. К ач марч ик , В. К илиан, П. Ст шель ч ак и Й. Щепк овск и. Hydrastor: масшт абируемое вт орич ное хранилище. В

ФАСТ , 2009.

1. К . Д убниц к ий, Л. Грыз , Л. Хелд т , М. К ач марч ик , В. К илиан, П. Ст шель ч ак , Й.

Щепк овск и, К . Унгуря ну и М. Велниц к и. HYDRAStor: масшт абируемое вт орич ное хранилище. В ФАСТ , 2009.

1. А. Эль -Шими, Р. К алач , А.К . Ад и, О.Д ж. Ли и С. Сенгупт а.

Первич ная д ед уплик ац ия д анных — к рупномасшт абное исслед ование и

проек т ирование сист емы. На ежегод ной т ехнич еск ой к онференц ии USENIX, 2012.

1. С. Хариз опулос, В. Ля н, DJ Абад и и С. Мэдд ен.

К омпромиссыпроиз вод ит ель ност и в баз ах д анных, опт имиз ированных д ля ч т ения . В VLDB, ст раниц ы487–498, 2006.

1. Б. Айер и Д . Уилхайт . Подд ержк а сжат ия д анных в баз ах д анных. 1994.
2. Н. Д жейн, М. Д алин и Р. Т евари. Taper: многоуровневый под ход к уст ранению из быт оч ност и при синхрониз ац ии реплик . В FAST, 2005.
3. E. Leonardi и SS Bhowmick. Xanadue: сист ема обнаружения из менений в XML-д анных в реля ц ионных баз ах д анных, не подд ерживающих д ревовид нуюст рук т уру. В SIGMOD, ст раниц ы1137–1140, 2007.
4. М. Лиллибрид ж, К . Эшг и, Д . Бхагват , В. Д еолалик ар,

G. Trezise и P. Camble. Раз реженное инд ек сирование: к рупномасшт абная

вст роенная д ед уплик ац ия с исполь з ованием выборк и и лок аль ност и. В FAST, 2009.

1. Д ж. Лелигер. К онт роль версий с помощь юgit: мощные инст румент ыи

мет од ыд ля совмест ной раз работ к и прог раммног о обеспеч ения . 2009.

1. Д ж. П. Мак д ональ д . Подд ержк а файловой сист емыд ля д ельт а-сжат ия .

Магист ерск ая д иссерт ац ия , К алифорнийск ий университ ет , Берк ли, 2000.

1. С. Мишра. Сжат ие д анных: ст рат егия , планирование емк ост и и луч шие прак т ик и. Т ехнич еск ая ст ать я SQL Server, 2009.
2. А. Мут ит ач ароен, Б. Ч ен и Д . Маз ь ер. А

Сет евая файловая сист ема с низ к ой пропуск ной способность ю. В SOSP, 2001.

1. Р. Паг и Ф. Ф. Род лер. К ук ушк ино хеширование. Журнал алгорит мов, 51(2):122–144, 2004.
2. М. Поэсс и Д . Пот апов. Сжат ие д анных в Oracle. В VLDB, ст раниц ы937–947, 2003.
3. Х. Пуч а, Д . Г. Анд ерсен и М. К аминск ий. Эк сплуат ац ия

сход ст во д ля з агруз ок из неск оль к их ист оч ник ов с исполь з ованием от печ ат к ов файлов.

В NSDI, 2007.

1. В. Пь ю. Списк и пропуск ов: вероя т ност ная альт ернат ива сбалансированным д еревь я . В Workshop on Algorithms and Data Structures, ст раниц ы437–449. Springer, 1989.
2. С. К уинлан и С. Д орвард . Venti: новый под ход к архивному хранению. В FAST, 2002.
3. Рабин М. О. Д ак т илоск опия случ айными полиномами.



13

### Machine Translated by Google

1. В. Раман, Г. Ат т алури, Р. Барбер, Н. Ч айнани, Д . К алмук , В. К уланд ай Сами, Д ж. Леенст ра, С. Лайт ст оун, С. Лю, Г. М.

Ломан и д р. Db2 с уск орением Blu: гораз д о боль ше, ч ем прост о хранилище ст олбц ов. VLDB, 6(11):1080–1091, 2013.

1. С. Сак р. Мет од ысжат ия XML: обз ор и

сравнение. Журнал к омпь ют ерных и сист емных наук , 75(5):303–322, 2009.

1. П. Шилан, М. Хуан, Г. Уоллес и В. Сюй.

WAN-опт имиз ированная реплик ац ия рез ервных наборов д анных с исполь з ованием пот ок овой д ельт а-к омпрессии. В FAST, 2012.

1. П. Шилан, Г. Уоллес, М. Хуан и В. Сюй. Д ель т а

сжат ое и д ед уплиц ированное хранилище с исполь з ованием пот ок овой лок аль ност и. USENIX Hot Storage, 2012.

1. К . Шринивасан, Т . Биссон, Г. Гуд сон и К . Воругант и.

idedup: Уч ит ывающая з ад ержк у вст роенная д ед уплик ац ия д анных д ля первич ного хранилища. В FAST, 2012.

1. М. Ст оунбрейк ер, DJ Абад и, А. Бат к ин, X. Ч ен,

М. Ч ерня к , М. Феррейра, Э. Лау, А. Лин, С. Мэдд ен, Э. О'Нил и д р. C- store: СУБД , ориент ированная на ст олбц ы. В VLDB, ст раниц ы553–564, 2005 г .

1. Т . Суэл и Н. Мемон. Алгорит мыд ель т а-сжат ия и уд аленной синхрониз ац ии файлов. Справоч ник по сжат июбез пот ерь , 2002.
2. Д . Т еод осиу, Ю. Гуревич , М. Манассе и Ж. Поркк а.

Опт имиз ац ия реплик ац ии файлов в сет я х с огранич енной пропуск ной способност ь юс исполь з ованием уд аленног о д ифференц иаль ного сжат ия . Т ехнич еск ий пред ст авит ель .

MSR-TR-2006-157, Microsoft Research, 2006.

1. WF Tichy. Rcs – сист ема к онт роля версий. Программное обеспеч ение: прак т ик а и опыт , 15(7):637–654, 1985.
2. Д . Т ренд афилов, Н. Мемон и Т . Суэль . zdelta: Эффек т ивный

Инст румент д ель т а-к омпрессии. Т ехнич еск ий от ч ет TR-CIS-2002-02, Полит ехнич еск ий университ ет , 2002.

1. А. Т рид желл. Эффек т ивные алгорит мысорт ировк и и синхрониз ац ия . В д ок т орск ой д иссерт ац ии, Авст ралийск ий нац иональ ный университ ет , 2000.
2. Г. Урд анет а, Г. Пь ер и М. Ван Ст ин. Вик ипед ия

Анализ рабоч ей нагруз к и д ля д ец ент рализ ованног о хост инга. К омпь ют ерные сет и, 53(11):1830–1845, 2009.

1. Y. Wang, DJ DeWitt и J.-Y. Cai. X-diff: Эффек т ивный алгорит м обнаружения из менений д ля XML-д ок умент ов. В ICDE, ст раниц ы519–530, 2003.
2. В. Ся , Х. Ц з я н, Д . Фэн и Ю. Хуа. Силос: А

Схема д ед уплик ац ии на основе сход ст ва-лок аль ност и с низ к ими из д ержк ами операт ивной памя т и и высок ой пропуск ной способность ю. На ежегод ной т ехнич еск ой к онференц ии USENIX, 2011.

1. Л. Сюй, А. Павло, С. Сенгупа, Д ж. Ли и Г. Р. Гангер.

Сок ращение пропуск ной способност и реплик ац ии д ля распред еленных баз д анных д ок умент ов. В SoCC, ст раниц ы222–235, 2015.

1. LL You, KT Pollack и DD Long. Глубок ое хранилище: архит ек т ура архивной сист емыхранения . В ICDE, ст раниц ы804–815, 2005.
2. Б. Ч жу, К . Ли и Р. Х. Пат т ерсон. Из бегание д иск а

уз к ое мест о в файловой сист еме д ед уплик ац ии д омена д анных. В FAST, 2008.

1. Д ж. З ив и А. Лемпель . Универсаль ный алгорит м д ля послед оват ель ного сжат ие д анных. Т руд ыIEEE по т еории информац ии, 23(3):337–343, 1977.
2. М. Жук овск и, С. Хеман, Н. Нес и П. Бонц . Суперск аля рное сжат ие к эша RAM- CPU. В ICDE, ст раниц ы59–59, 2006.



14