Дедупликация БД

13 июня 2025 г.

1 АННОТАЦИЯ

Статья рассматривает вопросы дедупликации на основе сходства для систем управления базами данных (СУБД) в режиме реального времени методом однопроходного кодирования:

- сжатие отдельных страниц базы данных
- сжатие сообщений журнала операций (oplog) на уровне блоков
- дельта-кодирование отдельных записей в базе данных на уровне байтов.

Преимущества метода однопроходного кодирования:

- уменьшение размера данных, хранящихся на диске, по сравнению с традиционными схемами сжатия
- уменьшение объема данных, передаваемых по сети для служб репликации.

Чтобы оценить работу алгоритма:

- написана модель распределенной NoSQL СУБД с поддержкой dedup в распределенную NoSQL СУБД
- проанализированы свойства модели.

Цель:

- получить значение коэффициента блочного сжатия
- получить значение коэффициента сжатия для дедупликации
- получить значение коэффициента сжатия для дедупликации с блочным сжатием

2 ВВЕДЕНИЕ

Темпы роста объема данных dS_{DB} превышают планы по закупкам вычислительнных мощностей организаций dS_{Plan} из-за медленного, чем ожидалось снижения стоимости оборудования:

$$\frac{dS_{DB}}{dt} \geq \frac{dS_{Plan}}{dt} \tag{1}$$

Применение фуннкции сжатия баз данных $C(S)$ является одним из решений этой проблемы - уменьшением хранимого размера БД S_{DB}:

$$C(S_{DB}) \leq S_{DB} \tag{2}$$

, настолько, что:

$$\frac{dC(S_{DB})}{dt} \leq \frac{dS_{Plan}}{dt} \tag{3}$$

Для хранения баз данных, помимо экономии места, сжатие сокращает количество страниц P_{FS}, на которых размещаются данные, что помогает:

- сократить количество дисковых операций ввода-вывода N_{Disk}

$$\frac{dN_{Disk}(C(S_{DB})}{dt} \leq \frac{dN_{Disk}(S_{DB})}{dt} \tag{4}$$

- повысить производительность

$$\frac{dP_{FS}(C(S_{DB})}{dt} \leq \frac{dP_{FS}(S_{DB})}{dt} \tag{5}$$

Для распределенных баз данных, реплицированных по географическим регионам, также существует острая необходимость в сокращении объема передачи данных S_{Net}, используемого для синхронизации реплик:

$$\frac{dS_{Net}(C(S_{DB})}{dt} \leq \frac{dS_{Net}(S_{DB})}{dt} \tag{6}$$

Наиболее широко используемый подход к сокращению объема данных в операционных СУБД — это сжатие на уровне блоков [1], [2] .

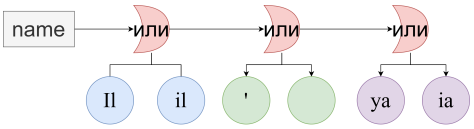
2.1 Эксперимент 1

Рассмотрим нормальзованные данные:

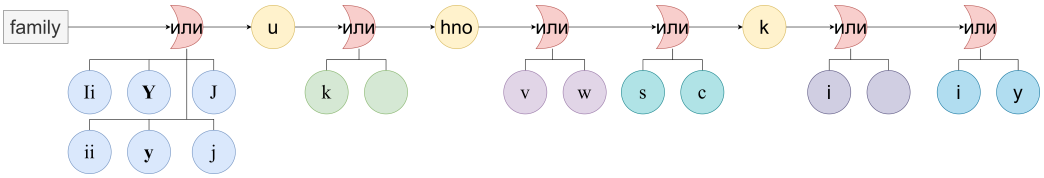
- Варианнты написания имени - `db/initial/name.dat` размер - 50Б
- Варианты написания фамилии - `db/initial/family.dat` размер - 4670Б

Те же данные в графовом представлении, используя DSL:

- Варианнты написания имени - `db/graph/name.grd` размер - 29Б



- Варианты написания фамилии - `db/graph/family.grd` размер - 85Б



Сожмем файлы и посмотрим на размер данных после сжатия:

```
./src/utils/run.bat
```

Сожмем построчно текстовые файлы:

```
./src/utils/gzip/run.bat
```

Размеры файлов приведены в таблице 1

Из результатов видно, что сжатие на уровне блоков (построчно) не решает проблему избыточности между блоками и, следовательно, оставляет значительные возможности для улучшения сжатия, например с помощью графов. В нашем случае, графы построены по байтно, поэтому сжатие не дает никаких результатов. Графы дедуплицируют данные, поэтому мы получаем максимальнный коэффициент сжатия - 55,59.

Дедупликация стала популярной в системах резервного копирования для устранения дублирующегося контента во всем корпусе данных, часто достигая гораздо более высоких коэффициентов сжатия. Поток резервного копирования делится на фрагменты, и в качестве идентификатора каждого фрагмента используется устойчивый к коллизиям хэш (например, SHA-1). Система дедупликации поддерживает глобальный индекс

Представление Даные	name	family	K_{name}	K_{family}
Текст (не сжатые)	50	4670	-	-
Текст (gzip файл)	34	813	1,47	5,74
Текст (gzip построчо)	108	7192	0,46	0,65
Граф (не сжатые)	29	85	1,72	54,94
Граф (gzip файл)	32	84	1,56	55,59
Граф (gzip построчно)	73	196	0,69	23,83

Таблица 1: Размер данных, где K_{name}, K_{family} - коэффициеннты сжатия

всех хэшей и использует его для обнаружения дубликатов. Дедупликация хорошо работает как для основных, так и для резервных наборов данных, которые состоят из больших файлов, которые редко изменяются (а если и изменяются, то изменения редки).

К сожалению, традиционные схемы дедупликации на основе фрагментов не подходят для операционных СУБД, где приложения выполняют запросы на обновление, которые изменяют 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 рассмотрим дедупликацию на основе сходства [3] для сжатия отдельных записей OLTP баз данных.

Вместо индексации каждого хеша фрагмента, алгоритм выбирает небольшое подмножество хешей фрагментов для каждой новой записи базы данных, а затем использует этот образец для 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 1.

Авторы объединили несколько методов для достижения этой эффективности:

- двустороннее кодирование для эффективной передачи закодированных новых записей (прямое кодирование) в удаленные реплики,
- сохраняя новые записи с закодированными формами выбранных

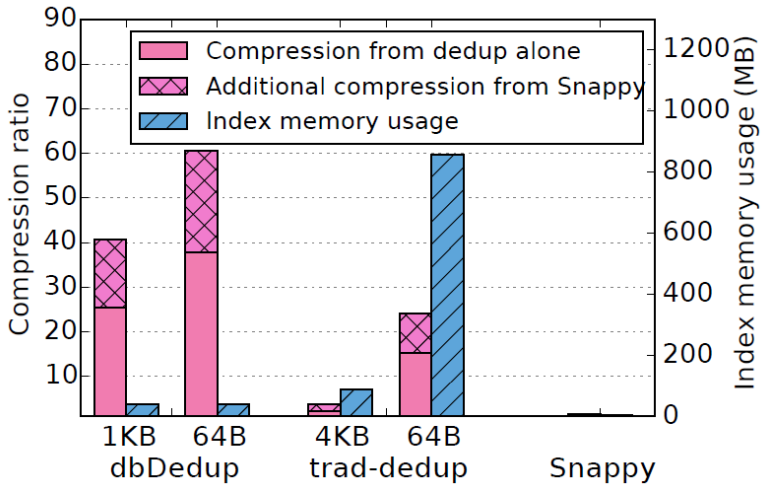


Рис. 1: Коэффициент сжатия и использование памяти индекса для данных Википедии, хранящихся в пяти конфигурациях MongoDB: с dbDedup (размер фрагмента 1 КБ и 64 Б), с традиционной дедупликацией (4 КБ и 64 Б) и с Snappy (сжатие на уровне блоков). dbDedup обеспечивает более высокую степень сжатия и меньшие накладные расходы на память индекса, чем традиционная дедупликация. Snappy обеспечивает такое же сжатие 1,6 для данных после дедупликации или исходных данных. [4]

исходных записей (обратное кодирование).

Список литературы

[1] G. V. Cormack. Data compression on a database system. Communications of the ACM, 28(12):1336-1342, 1985.

[2] B. Iyer and D. Wilhite. Data compression support in databases. 1994.

[3] L. Xu, A. Pavlo, S. Sengupa, J. Li, and G. R. Ganger. Reducing replication bandwidth for distributed document databases. In SoCC, pages 222–235, 2015.

[4] Xu, L., Pavlo, A., Sengupta, S., and Ganger, G. R.(2017). Online Deduplication for Databases. In Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD 17) (pp. 1355-1368). ACM Digital Library