Санкт-Петербургский государственный университет

Кафедра системного программирования

Разработка инструмента для сравнения алгоритмов дедупликации

Олег Антонович Пилецкий, группа 21.Б10-мм

Научный руководитель: к.ф.-м.н. В.И.Гориховский, доцент кафедры системного программирования

Санкт-Петербург
2024

Введение

- Огромные объемы данных
- Дедупликация — устранение повторений
  - разбиение на чанки;
  - хеширование;
  - удаление повторений;
  - сохранение;
- Очень много способов разбиения файлов на чанки
- Очень мало стандартизованных систем сравнения
  - DedupBench (2023, C++) — измеряет только чистую скорость

Требования к экспериментальному стенду

- Стенд — in-process файловая система с возможностью выбора алгоритма чанкинга
- Основные операции — запись и чтение больших файлов
- Встроенный бенчмаркинг скорости чтения/записи и коэффициента дедупликации
- Взаимозаменяемость алгоритмов хэширования и способов хранения данных
- На Rust (YADRO)

Требования к поддерживаемым методам дедупликации

- Content Defined Chunking (CDC)
  - Семейство алгоритмов
  - Делит массив данных на чанки в зависимости от содержимого
  - Обзор проводился в прошлом семестре
- Методы оптимизации
  1. Frequency Based Chunking (FBC)
     - ★ берет полученные алгоритмом CDC чанки;
     - ★ использует информацию о частоте чанков;
  2. Similarity Based Chunking (SBC)
     - ★ берет полученные алгоритмом CDC чанки;
     - ★ использует информацию о похожести чанков;
     - ★ хранит чанки-исходники и чанки-разницы;

Постановка задачи

Целью работы является создание стенда файловой системы на языке Rust, проводящего бенчмаркинг алгоритмов дедупликации и их оптимизаций

Задачи:

- провести обзор особенностей существующих файловых систем с открытым исходным кодом, поддерживающих дедупликацию;
- разработать архитектуру, поддерживающую конфигурируемость дедупликации;
- реализовать стенд, поддерживающий разные алгоритмы CDC;
- встроить поддержку систем оптимизации дедупликации FBC и SBC;
- провести первичное сравнение эффективности алгоритмов дедупликации;

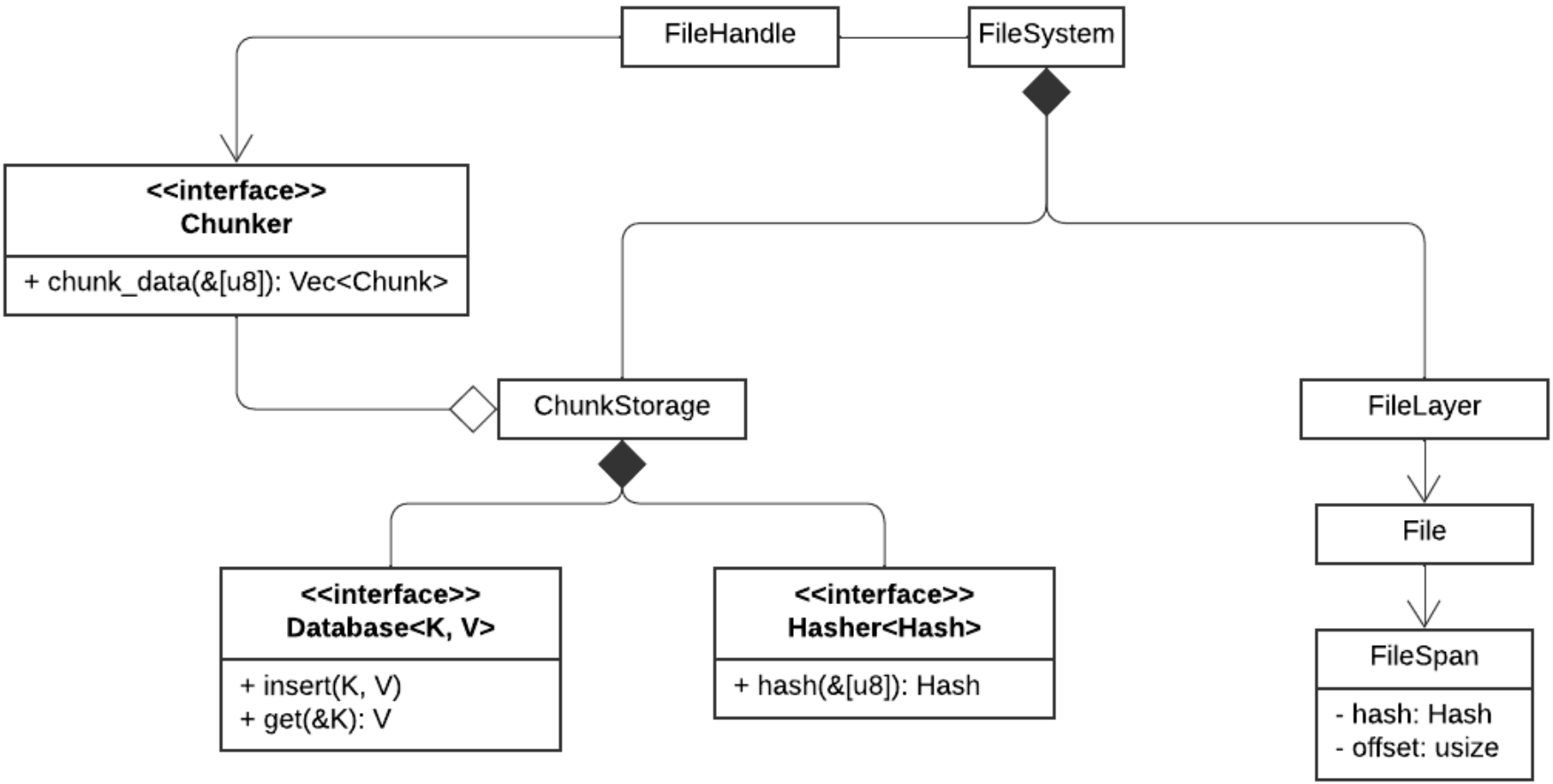
ZboxFS

- Написана на языке Rust
- Встроенная в приложение файловая система
- Алгоритм чанкинга — RabinCDC
- Чанки хранятся для каждого файла отдельно
- Чтение и запись при помощи встроенных интерфейсов
- Исследования в прошлом семестре

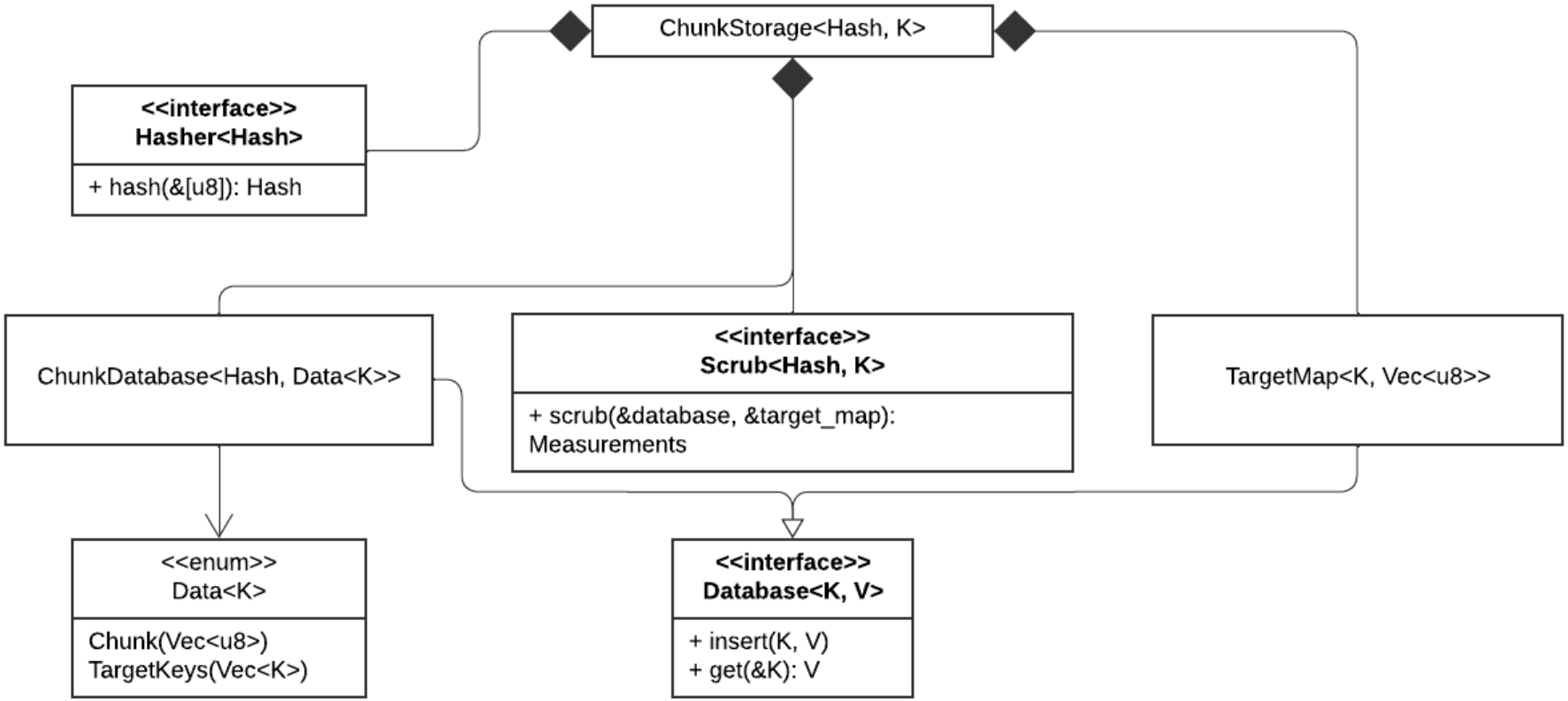
Restic

- Бэкап-система, написанная на Go
- Управление через интерфейс командной строки
- Данные хранятся в блобах — набор байт с соответствующим SHA-256 хэшем
- Используется RabinCDC
- При модификации сохраняются лишь модифицированные блобы
- Блобы привязаны к репозиторию

Архитектура



Поддержка методов оптимизации



Research Questions

1. Насколько результаты бенчмаркинга воспроизводимы на стенде?
2. Какой алгоритм обеспечивает лучшие характеристики?

Дизайн эксперимента

- Метрики для CDC
  - Коэффициент дедупликации (размер данных до / размер данных после)
  - Средний размер чанка
  - Скорость работы
- Метрики для методов оптимизации
  - Время работы
  - Количество просмотренных данных
  - Сколько данных осталось
- Интеграционные метрики
  - Скорость записи и чтения
- Наборы данных
  - Образы ядра Linux версий с 6.0 по 6.3, упакованные в tar-файл
  - Образ Ubuntu 22.04

Первичный эксперимент

	RabinCDC	UltraCDC	SuperCDC	LeapCDC
Linux	9.74 ± 0.09	7.46 ± 0.11	6.88 ± 0.14	6.63 ± 0.12
Ubuntu	2.37 ± 0.04	1.96 ± 0.07	1.52 ± 0.08	1.61 ± 0.03

Таблица: Время записи файла в систему, секунды

Результаты

В результате работы:

- были выявлены особенности файловых систем, такие как общее хранилище данных и чанки с переменной длиной
- был реализован стенд файловой системы, были встроены алгоритмы RabinCDC, LeapCDC, UltraCDC, SuperCDC
- была встроена поддержка FBC и SBC и она была проверена на наивной реализации SBC
- было проведено первичное исследование эффективности алгоритмов, показавшая, что стенд годится для сравнения методов дедупликации; по выбранным датасетам лучше всего себя показали SuperCDC и LeapCDC

Исходный код стенда файловой системы доступен на GitHub[1].

[1] `https://github.com/Piletskii-Oleg/chunkfs`