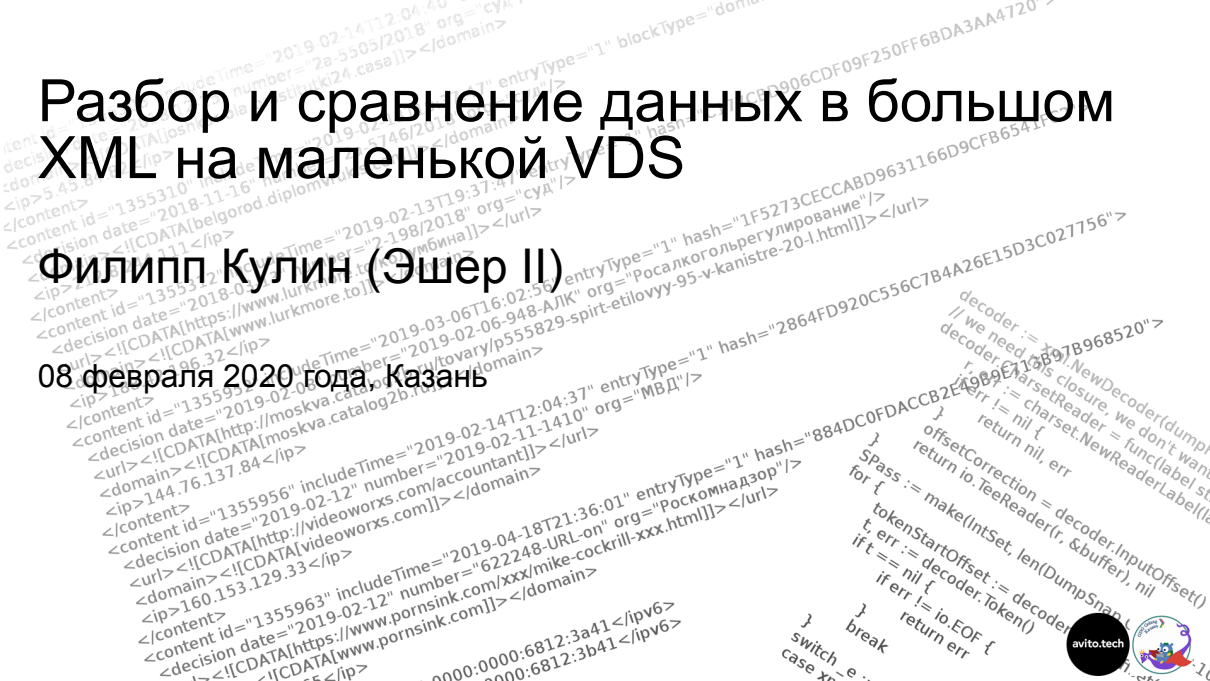
Разбор и сравнение данных в большом XML на маленькой VDS

Филипп Кулин (Эшер II)

08 февраля 2020 года, Казань



Откиньтесь на спинку кресла

- Эта презентация сделана с помощью LaTeX[1]
- Избежать большого количества кода не удалось
- Материал основан на работающем сервисе[3]
- Значительная часть кода написана сообществом

Задача

Что надо было сделать

- Разобрать XML-файл в 160Mb и положить в базу
- Обновлять базу по новым XML-файлам
- Предоставить интерфейс для доступа к базе

Задача

Что надо было сделать

- Разобрать XML-файл в 160Mb и положить в базу
- Обновлять базу по новым XML-файлам
- Предоставить интерфейс для доступа к базе

Условия

- Скорость разбора — единицы минут
- Недорогой виртуальный сервер
- Приоритет стандартных решений

Интересная задача

- **Никаких инноваций и ”rocket science”**
- Навязанные условия
- Чувствительность к работе памяти
- Чувствительность к ресурсам
- Хрестоматийные решения

Гадание

- Не всегда очевидны «тонкие» места
- Абстракции — «тихие омуты»

Гадание

- Не всегда очевидны «тонкие» места
- Абстракции — «тихие омуты»
- Тесты, бенчмарки, профилирование

Архитектура

- База данных
- Индексы по значениям для поиска
- Функционал наполнения базы из исходных данных
- Функционал обновления данных
- Функционал поиска данных

Формат исходных данных. XML

- 250 тысяч элементов `content`
- Размер каждого от сотни байт до **6**MB
- Общий размер данных свыше **150**MB

```
<?xml version="1.0" encoding="windows-1251"?>
<reg:register ...>
        <content id="656" ...> ... </content>
        ...
        <content id="2062369" ...> ... </content>
</reg:register>
```

Формат исходных данных. XML

- 250 тысяч элементов `content`
- Размер каждого от сотни байт до **6**MB
- Общий размер данных свыше **150**MB
- XML в кодировке CP1251

```
<?xml version="1.0" encoding="windows-1251"?>
<reg:register ...>
        <content id="656" ...> ... </content>
        ...
        <content id="2062369" ...> ... </content>
</reg:register>
```

Элемент Content

```
<content id="680741"...>
        <decision .../>
        <domain><![CDATA[example.com]]></domain>
        <url><![CDATA[https://example.com/smt]]></url>
        ...
        <ip>10.0.0.1</ip>
        ...
        <ipv6>fc00::beef</ipv6>
        ...
        <ipSubnet>10.1.0.0/16</ipSubnet>
        ...
        <ipSubnet6>fd00::/48</ipSubnet6>
        ...
</content>
```

- IP-адреса могут быть поштучно тысячами

Потоковый разбор XML

```
import ”encoding/xml”
...
decoder := xml.NewDecoder(dumpFile)
decoder.CharsetReader = charset.NewReaderLabel
for {
        t, err := decoder.Token()
        ...
        switch _e := t.(type) {
        case xml.StartElement:
                switch _e.Name.Local {
                case ”content”:
...
```

Потоковый разбор XML

- Создаем декодер

```
import ”encoding/xml”
...
decoder := xml.NewDecoder(dumpFile)
decoder.CharsetReader = charset.NewReaderLabel
for {
        t, err := decoder.Token()
        ...
        switch _e := t.(type) {
        case xml.StartElement:
                switch _e.Name.Local {
                case ”content”:
...
```

Потоковый разбор XML

- Создаем декодер
- Не забываем про кодировку

```
import ”encoding/xml”
...
decoder := xml.NewDecoder(dumpFile)
decoder.CharsetReader = charset.NewReaderLabel
for {
        t, err := decoder.Token()
        ...
        switch _e := t.(type) {
        case xml.StartElement:
                switch _e.Name.Local {
                case ”content”:
...
```

Потоковый разбор XML

- Создаем декодер
- Не забываем про кодировку
- Ловим каждый элемент

```
import "encoding/xml"
...
decoder := xml.NewDecoder(dumpFile)
decoder.CharsetReader = charset.NewReaderLabel
for {
        t, err := decoder.Token()
        ...
        switch _e := t.(type) {
        case xml.StartElement:
                switch _e.Name.Local {
                case "content":
...
```

Выбор формы хранения данных

- Время разбора XML — 10-200 секунд

Выбор формы хранения данных

- Время разбора XML — 10-200 секунд
- База нужна только для хранения

Выбор формы хранения данных

- Время разбора XML — 10-200 секунд
- База нужна только для хранения
- bbolt — заполнение час+

Выбор формы хранения данных

- Время разбора XML — 10-200 секунд
- База нужна только для хранения
- bbolt — заполнение час+
- map + скорость разбора = **победа**

Формат исходных данных. XML

- 250 тысяч элементов `content`
- Размер каждого от сотни байт до **6**MB
- Общий размер данных свыше **150**MB

```
<?xml version="1.0" encoding="windows-1251"?>
<reg:register ...>
        <content id="656" ...> ... </content>
        ...
        <content id="2062369" ...> ... </content>
</reg:register>
```

Формат исходных данных. XML

- 250 тысяч элементов `content`
- Размер каждого от сотни байт до **6**MB
- Общий размер данных свыше **150**MB
- XML в кодировке CP1251

```
<?xml version="1.0" encoding="windows-1251"?>
<reg:register ...>
        <content id="656" ...> ... </content>
        ...
        <content id="2062369" ...> ... </content>
</reg:register>
```

Формат хранения данных

- Каждый `content` имеет уникальный числовой `id`
- `id` очень разреженные

Формат хранения данных

- Каждый `content` имеет уникальный числовой `id`
- `id` очень разреженные
- Решение: `map[int]*TContent`
  `TContent` — структура для `DecodeElement()`

```
case "content":
        err := decoder.DecodeElement(&v, &_e)
```

Хранение данных. Грабли

- Уменьшаем аллокации

Хранение данных. Грабли

- Уменьшаем аллокации
- Улучшательство: `map[int]TContent`

Хранение данных. Грабли

- Уменьшаем аллокации
- Улучшательство: `map[int]TContent`
- **Боль**

Хранение данных. Грабли

- Уменьшаем аллокации
- Улучшательство: `map[int]TContent`
- **Боль** программы ощущалась физически...
- Расширение карты — очень дорогая операция
- Всё-таки `map[int]*TContent`

Хранение данных. Оптимизация

- Часть элементов имеет аттрибут `ts`:

```
ts="2020-02-06T19:54:00+03:00"
```

- IPv4-адреса представлены элементом IP:

```
<ip ts="2020-02-06T19:54:00+03:00">
        10.0.0.1
</ip>
```

Хранение данных. Оптимизация

- Часть элементов имеет аттрибут `ts`:

```
ts="2020-02-06T19:54:00+03:00"
```

- IPv4-адреса представлены элементом IP:

```
<ip ts="2020-02-06T19:54:00+03:00">
        10.0.0.1
</ip>
```

- Две строки против двух целых чисел
- Миллионы структур с IPv4-адресами

Оптимизация конвертации данных

- Преобразование стандартными средствами

```
ip := net.ParseIP(s)
intIp =: binary.BigEndian.Uint32(ip[12:16])
```

Оптимизация конвертации данных

- Преобразование стандартными средствами

```
ip := net.ParseIP(s)
intIp =: binary.BigEndian.Uint32(ip[12:16])
```

- Пишем менее универсально, но без аллокаций

Оптимизация конвертации данных

- Преобразование стандартными средствами

```
ip := net.ParseIP(s)
intIp =: binary.BigEndian.Uint32(ip[12:16])
```

- Пишем менее универсально, но без аллокаций
- Сравниваем

```
Benchmark_Standart-4   4295910   268    ns/op   48 B/op   3 allocs/op
Benchmark_Custom-4    18941194    60.8 ns/op    0 B/op   0 allocs/op
```

Более детальное преобразование

- Наполняем `TContent` «вручную»

```
case "content":
err := decoder.DecodeElement(&v, &_e)
% err := UnmarshalContent(tempBuf, &v)
...
func UnmarshalContent(b []byte, v *TContent) error {
    buf := bytes.NewReader(b)
    decoder := xml.NewDecoder(buf)
    ...
    case elementIp:
        ip := TXMLIp{}
        if err := decoder.DecodeElement(&ip, &element); err != nil {
```

Более детальное преобразование

- Наполняем `TContent` «вручную»
- Значительное уменьшение потребления памяти

```
case "content":
% err := decoder.DecodeElement(&v, &_e)
err := UnmarshalContent(tempBuf, &v)
...
func UnmarshalContent(b []byte, v *TContent) error {
    buf := bytes.NewReader(b)
    decoder := xml.NewDecoder(buf)
    ...
    case elementIp:
        ip := TXMLIp{}
        if err := decoder.DecodeElement(&ip, &element); err != nil {
```

Хранение данных. Индексы

- Самая простая и скучная часть
- map нужных данных
- Значения — массивы `id` элементов `content`

Обновление данных

- Упрощаем до сравнения `content` целиком

Обновление данных

- Упрощаем до сравнения `content` целиком
- Считаем контрольные суммы

Контрольные суммы

- Считал SHA256

Контрольные суммы

- Считал SHA256
- Профилирование показало, что выбор неудачный

Контрольные суммы

- Считал SHA256
- Профилирование показало, что выбор неудачный
- Выбрал на 64-bit FNV-1

Контрольные суммы

- Считал SHA256
- Профилирование показало, что выбор неудачный
- Выбрал на 64-bit FNV-1
  - Проверил, заглянув «под капот»

Как считаем контрольные суммы

- Преобразование `TContent` обратно в XML

Как считаем контрольные суммы

- Преобразование `TContent` обратно в XML
  - Требует двойного преобразования **каждого** элемента

Как считаем контрольные суммы

- Преобразование `TContent` обратно в XML
  - Требует двойного преобразования **каждого** элемента
- Использование `io.TeeReader`

Как считаем контрольные суммы

- Преобразование `TContent` обратно в XML
  - Требует двойного преобразования **каждого** элемента
- Использование `io.TeeReader`
  - Декодируем только изменившиеся элементы `content`

TeeReader и Decoder. Грабли №1

```
tr := io.TeeReader(dumpFile, &buffer)
decoder := xml.NewDecoder(tr)
decoder.CharsetReader = charset.NewReaderLabel
for {
        tokenStartOffset := decoder.InputOffset()
```

TeeReader и Decoder. Грабли №1

```
tr := io.TeeReader(dumpFile, &buffer)
decoder := xml.NewDecoder(tr)
decoder.CharsetReader = charset.NewReaderLabel
for {
        tokenStartOffset := decoder.InputOffset()
```

- ... не работает, данные не синхронны

TeeReader и Decoder. Грабли №1

```
tr := io.TeeReader(dumpFile, &buffer)
decoder := xml.NewDecoder(tr)
decoder.CharsetReader = charset.NewReaderLabel
for {
        tokenStartOffset := decoder.InputOffset()
```

- ... не работает, данные не синхронны
- XML декодер «шагает» по UTF-8 строке
- контрольные суммы «шагают» по CP1251 строке

TeeReader и Decoder. Грабли №2

```
decoder := xml.NewDecoder(dumpFile)
decoder.CharsetReader =
        func(l string, i io.Reader) (io.Reader, error) {
        r, err := charset.NewReaderLabel(l, i)
        return io.TeeReader(r, &buffer), nil
}
for {
        tokenStartOffset := decoder.InputOffset()
```

TeeReader и Decoder. Грабли №2

```
decoder := xml.NewDecoder(dumpFile)
decoder.CharsetReader =
        func(l string, i io.Reader) (io.Reader, error) {
        r, err := charset.NewReaderLabel(l, i)
        return io.TeeReader(r, &buffer), nil
}
for {
        tokenStartOffset := decoder.InputOffset()
```

- ... не работает, данные не синхронны

TeeReader и Decoder. Грабли №2

```
decoder := xml.NewDecoder(dumpFile)
decoder.CharsetReader =
        func(l string, i io.Reader) (io.Reader, error) {
        r, err := charset.NewReaderLabel(l, i)
        return io.TeeReader(r, &buffer), nil
}
for {
        tokenStartOffset := decoder.InputOffset()
```

- ... не работает, данные не синхронны
- XML декодер «перепрыгивает» через заголовок
- контрольные суммы не «перепрыгивают»

TeeReader и Decoder

Заработало!

```
decoder := xml.NewDecoder(dumpFile)
decoder.CharsetReader =
        func(l string, i io.Reader) (io.Reader, error) {
        r, err := charset.NewReaderLabel(l, i)
        offsetCorrection = decoder.InputOffset()
        return io.TeeReader(r, &buffer), nil
}
for {
        tokenStartOffset := decoder.InputOffset() -
                offsetCorrection
```

Отличное наглядное упражнение

- Ридеры/райтеры — go-way
- I/O через буфер тянется ещё с 90-ых
- Ридеры/райтеры на интерфейсах — новое в go
- У новичков затруднено понимание этой абстракции
- Хороший пример использования «замыкания»
- Разобранная задача — отличное упражнение

Подключаем gRPC

- Универсальное простое рабочее решение

Подключаем gRPC

- Универсальное простое рабочее решение
- Данные хранить сразу в формате gRPC

Подключаем gRPC

- Универсальное простое рабочее решение
- Данные хранить сразу в формате gRPC
  - gRPC пытается всё хранить ссылками

Подключаем gRPC

- Универсальное простое рабочее решение
- Данные хранить сразу в формате gRPC
  - gRPC пытается всё хранить ссылками
  - Несовместимо с нашей борьбой со ссылками

Делаем промежуточный тип

- Из XML преобразуем в `TContent`

Делаем промежуточный тип

- Из XML преобразуем в `TContent`
- `TContent` пакуем в json

Делаем промежуточный тип

- Из XML преобразуем в `TContent`
- `TContent` пакуем в json
- Новый тип `TMinContent` содержит:
  - метаданные
  - данные для сравнения (и индекса)
  - контрольную сумму для сравнения
  - Полные данные `TContent` в виде json

Делаем промежуточный тип

- Из XML преобразуем в `TContent`
- `TContent` пакуем в json
- Новый тип `TMinContent` содержит:
  - метаданные
  - данные для сравнения (и индекса)
  - контрольную сумму для сравнения
  - Полные данные `TContent` в виде json
- В базу кладем `TMinContent`

Делаем промежуточный тип

- Из XML преобразуем в `TContent`
- `TContent` пакуем в json
- Новый тип `TMinContent` содержит:
  - метаданные
  - данные для сравнения (и индекса)
  - контрольную сумму для сравнения
  - Полные данные `TContent` в виде json
- В базу кладем `TMinContent`
- Увеличен общий объём данных из-за дублирования

Данные gRPC

- Массив «сообщений». «Сообщение» содержит:
  - метаданные
  - полные данные в виде json-строки с `TContent`

Данные gRPC

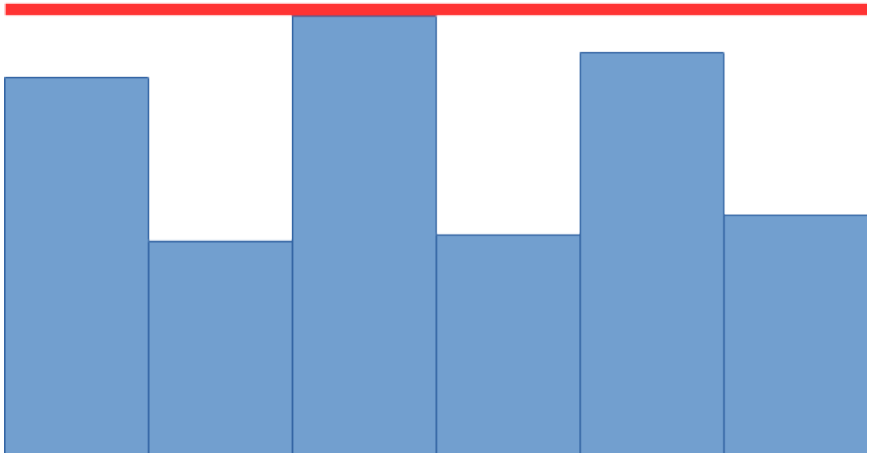
- Массив «сообщений». «Сообщение» содержит:
  - метаданные
  - полные данные в виде json-строки с `TContent`
- Буфер под отдаваемые данные минимален

Данные gRPC

- Массив «сообщений». «Сообщение» содержит:
  - метаданные
  - полные данные в виде json-строки с `TContent`
- Буфер под отдаваемые данные минимален
- *Можно написать свою реализацию gRPC*

Что ещё можно «подкрутить»?

Использование памяти



Нас интересует только верхняя граница

Настройки рантайма

- Управление сборщиком мусора

```
% GOGC=50 /bin/myapp
```


Настройки рантайма

- Управление сборщиком мусора

```
% GOGC=50 /bin/myapp
```

  - Малоэффективно, «подтормаживает»

Настройки рантайма

- Управление сборщиком мусора

```
% GOGC=50 /bin/myapp
```

  - Малоэффективно, «подтормаживает»
- Возврат памяти системе

```
% GODEBUG=madvdontneed=1 /bin/myapp
```

Настройки рантайма

- Управление сборщиком мусора

```
% GOGC=50 /bin/myapp
```

  - Малоэффективно, «подтормаживает»
- Возврат памяти системе

```
% GODEBUG=madvdontneed=1 /bin/myapp
```

  - Малоэффективно в пике, «тормозит»

Настройки рантайма

- Управление сборщиком мусора

```
% GOGC=50 /bin/myapp
```

  - Малоэффективно, «подтормаживает»
- Возврат памяти системе

```
% GODEBUG=madvdontneed=1 /bin/myapp
```

  - Малоэффективно в пике, «тормозит»

Чем хуже код — тем больше негативный эффект

Итог

- Данные в map в памяти
- map для индексов
- Хранения на диске нет
- Потоковый разбор XML с TeeReader
- Оптимизация форматов данных
- Сравнение через контрольные суммы
- Вспомогательный тип для хранения данных

Вопросы

Избави, Боже, нас от ярости норманнов
и «интересных задач»

schors@gmail.com

Ссылки

[1] Филипп Кулин. *Пишем презентации в LaTeX*. https://habr.com/ru/post/471352/.

[2] *Мониторинг реестра запрещенных сайтов*. https://usher2.club/.

[3] *Исходный код сервиса обработки выгрузки*. https://github.com/usher2/u2ckdump.

[4] *Исходный код Telegram-бота для проверки сайта*. https://github.com/usher2/u2ckbot.