Яндекс Такси

Пилорама

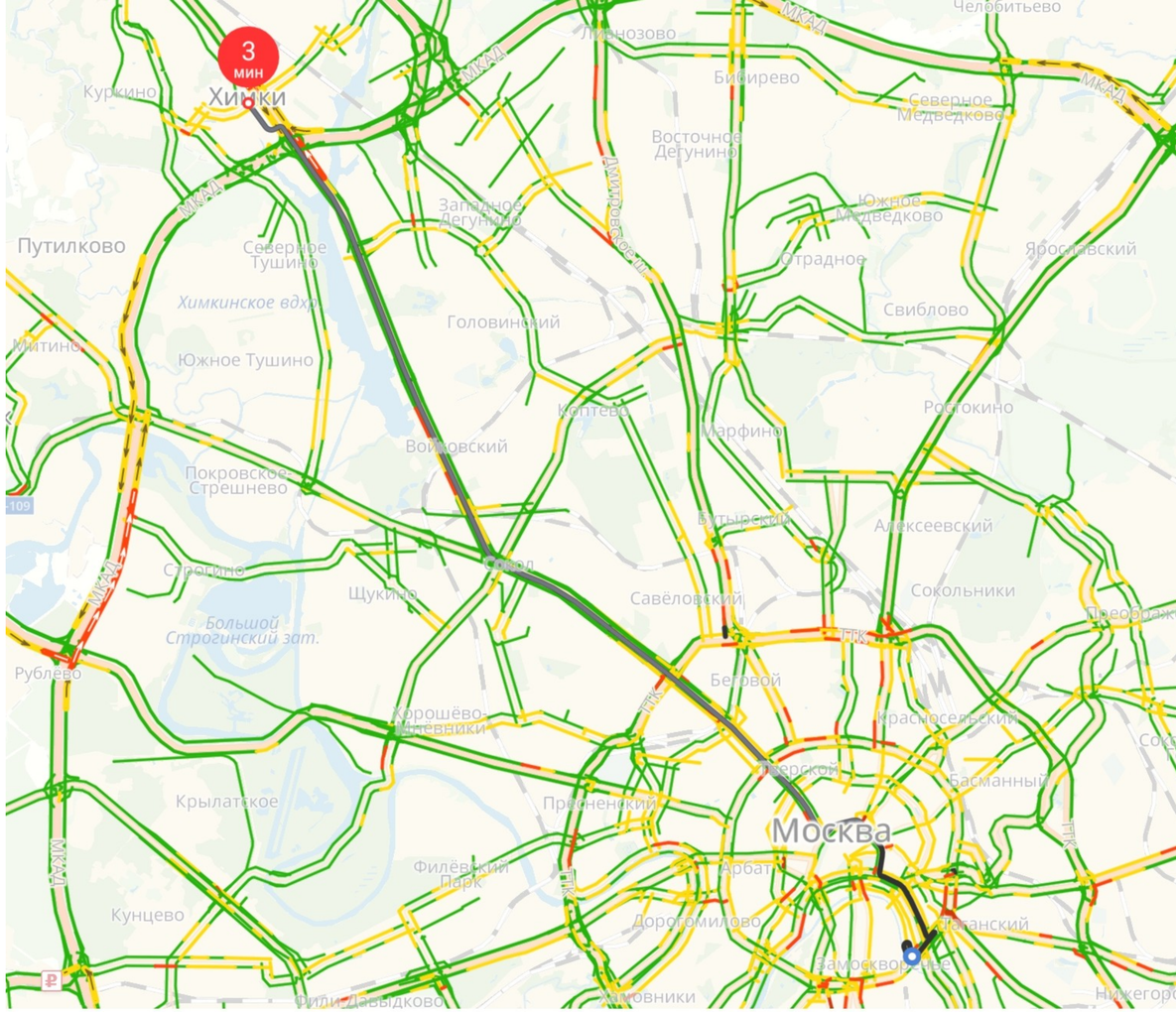
Полухин Антон

Antony Polukhin

Яндекс Такси

Содержание

- Java не тормозит
- Пилорама
  - C++
  - Радости сопрограмм
  - Проблемы с ОС



Немного про нашу нагрузку

(или о том как сделать больно известным решениям)

Нагрузка

1 микросервис в 1 ДЦ:

Нагрузка

1 микросервис в 1 ДЦ:

~ 20 000 событий в секунду

Нагрузка

1 микросервис в 1 ДЦ:

~ 20 000 событий в секунду

~ 30 GB логов в час

Нагрузка

1 микросервис в 1 ДЦ:

~ 20 000 событий в секунду

~ 30 GB логов в час

В сумме:

Нагрузка

1 микросервис в 1 ДЦ:

~ 20 000 событий в секунду

~ 30 GB логов в час

В сумме:

> 10^9 событий

Нагрузка

1 микросервис в 1 ДЦ:

~ 20 000 событий в секунду

~ 30 GB логов в час

В сумме:

> 10^9 событий

> 1 TB логов в час

Что с этими терабайтами данных делать?

Сохранять и анализировать :)

Logstash

– Программа для сбора, трансформации и складирования логов.

Logstash

– Программа для сбора, трансформации и складирования логов.

Бесплатное и очень популярное Open Source приложение на Java.

Logstash

– Программа для сбора, трансформации и складирования логов.

Бесплатное и очень популярное Open Source приложение на Java.

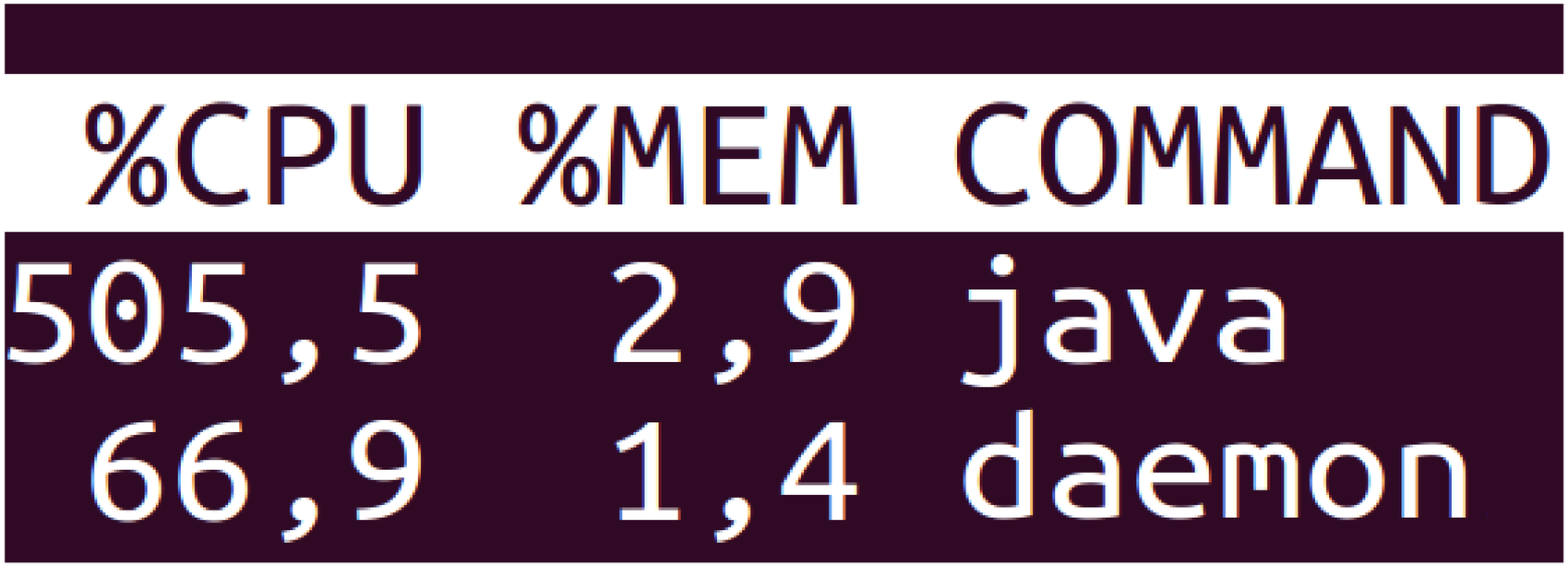
Что может пойти не так?

Logstash

– Программа для сбора, трансформации и складирования логов.

Бесплатное и очень популярное Open Source приложение на Java.

Что может пойти не так?



Что тормозит?

Наш pipeline

Наш pipeline

- Считываем данные с диска

Наш pipeline

- Считываем данные с диска
- Разбиваем на ключ-значение

Наш pipeline

- Считываем данные с диска
- Разбиваем на ключ-значение
- Применяем парочку простых правил

Наш pipeline

- Считываем данные с диска
- Разбиваем на ключ-значение
- Применяем парочку простых правил
- Формируем запись в новом формате

Наш pipeline

- Считываем данные с диска
- Разбиваем на ключ-значение
- Применяем парочку простых правил
- Формируем запись в новом формате
- Отсылаем в удалённое хранилище

Logstash

Вооружаемся perf...

... обнаруживаем что основная часть времени тратится в каких-то конкурентных ассоциативных контейнерах и ещё более странных местах.

Наш pipeline

- Считываем данные с диска
- Разбиваем на ключ-значение
- Применяем парочку простых правил
- Формируем запись в новом формате
- Отсылаем в удалённое хранилище

Сделаем свой logstash!

Сделаем свой logstash!

- Без лишнего

Сделаем свой logstash!

- Без лишнего
- Шустрый

Сделаем свой logstash!

- Без лишнего
- Шустрый
- Модный и современный

Пилорама

(A factory in which logs are sawed and send to remote)

Пилорама

- Считываем данные с диска
- Разбиваем на ключ-значение
- Применяем парочку простых правил
- Формируем запись в новом формате
- Отсылаем в удалённое хранилище

Пилорама

- Считываем данные с диска
- Разбиваем на ключ-значение
- Применяем парочку простых правил
- Формируем запись в новом формате
- Отсылаем в удалённое хранилище

Пилорама

- Считываем данные с диска
- Формируем запись в новом формате и с применёнными правилами
- Отсылаем в удалённое хранилище

Конвертация

```
utils::string_view chunk;
```

Конвертация

```
do {
  const std::string key = GetKey();
  if (state_ == State::kIncompleteRecord) {
    // Stop parsing to let the producer write more data and finish the record.
    return 0;
  }
  const utils::string_view value = GetValue();
  if (state_ == State::kIncompleteRecord) { return 0; }
  WriteWithFilters(writer, key, value);
} while (state_ == State::kParsing);
```

Конвертация

```
do {
  const std::string key = GetKey();
  if (state_ == State::kIncompleteRecord) {
    // Stop parsing to let the producer write more data and finish the record.
    return 0;
  }
  const utils::string_view value = GetValue();
  if (state_ == State::kIncompleteRecord) { return 0; }
  WriteWithFilters(writer, key, value);
} while (state_ == State::kParsing);
```

Конвертация

```
do {
  const std::string key = GetKey();
  if (state_ == State::kIncompleteRecord) {
    // Stop parsing to let the producer write more data and finish the record.
    return 0;
  }
  const utils::string_view value = GetValue();
  if (state_ == State::kIncompleteRecord) { return 0; }
  WriteWithFilters(writer, key, value);
} while (state_ == State::kParsing);
```

Конвертация

```
do {
  const std::string key = GetKey();
  if (state_ == State::kIncompleteRecord) {
    // Stop parsing to let the producer write more data and finish the record.
    return 0;
  }
  const utils::string_view value = GetValue();
  if (state_ == State::kIncompleteRecord) { return 0; }
  WriteWithFilters(writer, key, value);
} while (state_ == State::kParsing);
```

Конвертация

```
do {
  const std::string key = GetKey();
  if (state_ == State::kIncompleteRecord) {
    // Stop parsing to let the producer write more data and finish the record.
    return 0;
  }
  const utils::string_view value = GetValue();
  if (state_ == State::kIncompleteRecord) { return 0; }
  WriteWithFilters(writer, key, value);
} while (state_ == State::kParsing);
```

Конвертация

```
do {
  const std::string key = GetKey();
  if (state_ == State::kIncompleteRecord) {
    // Stop parsing to let the producer write more data and finish the record.
    return 0;
  }
  const utils::string_view value = GetValue();
  if (state_ == State::kIncompleteRecord) { return 0; }
  WriteWithFilters(writer, key, value);
} while (state_ == State::kParsing);
```

Скорость

Скорость

> 360к записей в секунду

> 75MB в секунду

Скорость

> 360к записей в секунду

> 75MB в секунду

> 270 GB в час

Скорость

> 360к записей в секунду

> 75MB в секунду

> 270 GB в час:

~10 микросервисов на 1ом ядре Пилорамы

~ Все логи Такси на 4х ядрах Пилорамы

А что так медленно?

(немного боли)

Эх... нам бы C++20 а не cctz

```
// Formats a std::tm using strftime(3).
void FormatTM(std::string* out, const std::string& fmt, const std::tm& tm) {
  for (int i = 2; i != 32; i *= 2) {
    size_t buf_size = fmt.size() * i;
    std::vector<char> buf(buf_size);
    if (size_t len = strftime(&buf[0], buf_size, fmt.c_str(), &tm)) {
      out->append(&buf[0], len);
      return;
    }
  }
}
```

Эх... нам бы C++20 а не cctz

```
// Formats a std::tm using strftime(3).
void FormatTM(std::string* out, const std::string& fmt, const std::tm& tm) {
  for (int i = 2; i != 32; i *= 2) {
    size_t buf_size = fmt.size() * i;
    std::vector<char> buf(buf_size);
    if (size_t len = strftime(&buf[0], buf_size, fmt.c_str(), &tm)) {
      out->append(&buf[0], len);
      return;
    }
  }
}
```

Пилорама

- Считываем данные с диска
- Формируем запись в новом формате и с применёнными правилами
- Отсылаем в удалённое хранилище

Пилорама

- boost::interprocess::mapped_region
- Формируем запись в новом формате и с применёнными правилами
- Отсылаем в удалённое хранилище

Пилорама

- Считываем данные с диска
- Формируем запись в новом формате и с применёнными правилами
- Отсылаем в удалённое хранилище

Отсылка

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    engine::SleepFor(1s);
    continue;
  }
  SendToRemote(result);
  result.clear();
} while (true);
```

Отсылка

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    engine::SleepFor(1s);
    continue;
  }
  SendToRemote(result);
  result.clear();
} while (true);
```

Отсылка

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    engine::SleepFor(1s);
    continue;
  }
  SendToRemote(result);
  result.clear();
} while (true);
```

Отсылка

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    engine::SleepFor(1s);
    continue;
  }
  SendToRemote(result);
  result.clear();
} while (true);
```

Sleep?

(вы точно пишете высокопроизводительный сервис?)

Как-то не похоже на высоко производительную вещь!

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    engine::SleepFor(1s);
    continue;
  }
  SendToRemote(result);
  result.clear();
} while (true);
```

Ой

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    engine::SleepFor(1s); // асинхронный метод
    continue;
  }
  SendToRemote(result);  // асинхронный метод
  result.clear();
} while (true);
```

Coroutines TS

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    co_await engine::SleepFor(1s); // асинхронный метод
    continue;
  }
  co_await SendToRemote(result);  // асинхронный метод
  result.clear();
} while (true);
```

Coroutines TS

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    co_await engine::SleepFor(1s); // асинхронный метод
    continue;
  }
  co_await SendToRemote(result);  // асинхронный метод
  result.clear();
} while (true);
```

Coroutines TS

```
std::string result;
do {
  AppendNewData(result);
  if (result.size() < treshold) {
    co_await engine::SleepFor(1s); // асинхронный метод
    continue;
  }
  co_await SendToRemote(result);  // асинхронный метод
  result.clear();
} while (true);
```

Немного о том, чего не хватает

Хотелось бы...

- C++20

Хотелось бы...

- C++20
- Стандартную библиотеку C++ побольше

Хотелось бы...

- C++20
- Стандартную библиотеку C++ побольше
- Правильный Async IO

Спасибо

Полухин Антон

Старший разработчик Yandex.Taxi

antoshkka@gmail.com

antoshkka@yandex-team.ru

https://github.com/apolukhin

C++ РГ21 C++ РОССИЯ https://stdcpp.ru/