# ОТЛАДКА И ЭКСПЛУАТАЦИЯ RAILS-ПРИЛОЖЕНИЙ

FUN BOX

DevConf

# ОБО МНЕ


Руководитель
направления разработки

ruby/erlang/nodejs

@goganchic

# СЛОЖНОСТИ ЭКСПЛУАТАЦИИ

- ОТЛАДКА

# СЛОЖНОСТИ ЭКСПЛУАТАЦИИ

- ОТЛАДКА
- ВЫСОКИЕ НАГРУЗКИ

# СЛОЖНОСТИ ЭКСПЛУАТАЦИИ

- ОТЛАДКА
- ВЫСОКИЕ НАГРУЗКИ
- ПРОФИЛАКТИКА

# ОРГАНИЗАЦИОННЫЕ РЕШЕНИЯ

# ТЕСТИРОВАНИЕ И КОД-РЕВЬЮ

Чем раньше найдена ошибка — тем она дешевле

Тесты — для продуманных заранее краевых случаев, код-ревью — для возможно пропущенных

Unit-тесты не отменяют приемочных

# ЗНАТЬ ИСПОЛЬЗУЕМЫЕ ИНСТРУМЕНТЫ

Код библиотек не идеален

Разные библиотеки могут конфликтовать друг с другом

Лучше не использовать то, что сложно понять

# МОНИТОРИНГ

Слежение за ресурсами
(CPU/RAM/HDD) — Zabbix

Слежение за внешними системами
(DB, Redis, внешние API) — Zabbix+скрипты

Внутренние метрики — PulseMeter

# PULSEMETER

```
sensors = PulseMeter::Sensor::Configuration.new(
 root_webpage_requests_per_5_minutes: {
   sensor_type: 'timelined/counter',
   args: {interval: 5.minutes, ttl: 1.week}
 }
)
PulseMeter::Observer.observe_method(HomeController, :index,
sensors) do
 root_webpage_requests_per_5_minutes(1)
end
```

# PULSEMETER


PulseMeter

https://github.com/savonarola/pulse-meter

# ОПОВЕЩЕНИЯ

Определить, что нормально

Выставить триггеры

Определить приоритеты оповещений

# МОНИТОРЬ ЗАРАНЕЕ

# ЛОГИ

Processing GET "/requests/10" for 1.1.1.1

Completed 200 OK in 336ms
(Views: 0.8ms | ActiveRecord: 12.9ms)

ПОСМОТРЮ-КА Я

ТВОИ ЛОГИ

# ЛОГИ

Processing GET "/requests/10" for 1.1.1.1
Completed 200 OK in 336ms (Views: 0.8ms | ActiveRecord: 12.9ms)

Когда был сделан запрос?
Как получить все логи по запросу?
Были ли запросы во внешние API и если были — то какие ответы были получены?
Для какого пользователя запрос?

# ЛОГИ

[2015-04-18 20:11:39] [7d1fe7a2] [user_id=42] Processing GET "/requests/10" for 1.1.1.1
[2015-04-18 20:11:39] [7d1fe7a2] [user_id=42] PartnerApi request:
http://example.com/?param=value
[2015-04-18 20:11:39] [7d1fe7a2] [user_id=42] PartnerApi response: 403 Forbidden
[2015-04-18 20:11:39] [7d1fe7a2] Completed 200 OK in 336ms (Views: 0.8ms |
ActiveRecord: 12.9ms)

+ консолидация логов (logstash или скрипты)

# АРХИТЕКТУРНЫЕ РЕШЕНИЯ

# +

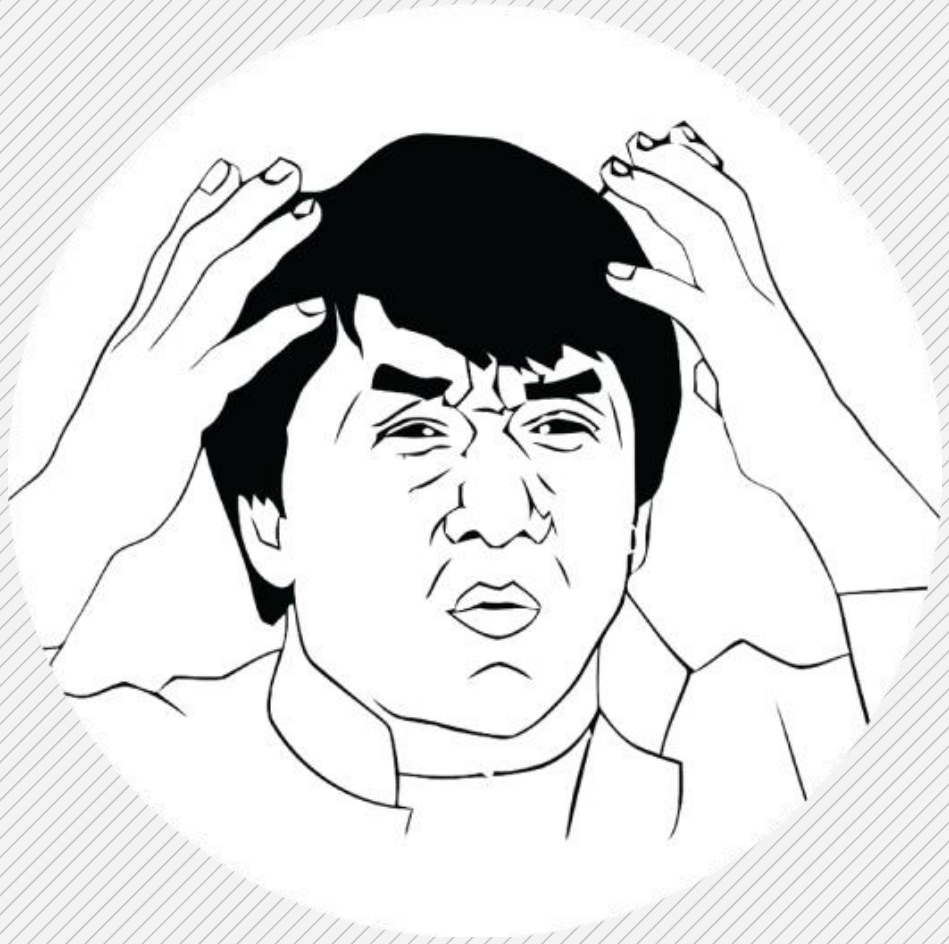
# ПОЛЕЗНЫЕ УТИЛИТЫ

# АСИНХРОННОСТЬ

```
def process_request(params)
 resp = HttpRequest.perform(params_for(request))
 data = ResponseParser.parse(resp)
 DataProcessor.process(data)
end
requests.each {|r| process_request(r)}
```

# АСИНХРОННОСТЬ — ЗЛО

```
def process_request(request)
 AsyncHttpRequest.perform(params_for(request)) do |resp|
   AsyncResponseParser.parse(resp) do |data|
     AsyncDataProcessor.process(data)
   end
 end
end
requests.each {|r| process_request(r)}
```



# АСИНХРОННОСТЬ — ЗЛО

Код менее читаемый

Все библиотеки должны знать про асинхронность

Все ошибки должны корректно обрабатываться
(т.к. общее состояние)

# МНОГОПОТОЧНОЕ ПРОГРАММИРОВАНИЕ

ТЕОРИЯ

РЕАЛЬНОСТЬ

# МНОГОПОТОЧНОСТЬ — БОЛЬ

Воспроизвести баг — нереально

Получить магию — проще простого

Чем выше нагрузка — тем хуже магия

# SEGMENTATION FAULTS


# SEGMENTATION FAULTS EVERYWHERE!

# БЫСТРЫЕ ТРАНЗАКЦИИ В БД

БД — общий ресурс

Заблокировал общий ресурс —
повесил всю систему

# ДЛИТЕЛЬНЫЕ ОПЕРАЦИИ — В ОЧЕРЕДЬ

HTTP обработчиков в Rails-проектах обычно мало

Заблокировал несколько обработчиков — повесил всю систему

# ПЛАВНАЯ ДЕГРАДАЦИЯ — ЭТО ХОРОШО

Возможно без некоторых внешних систем сервис может работать

# АКТУАЛЬНЫЕ ВЕРСИИ БИБЛИОТЕК

Современные системы делают из готовых компонентов

Возможно ваши баги уже кто-то поправил
bundle outdated в помощь

Избегайте пре-альфа версий

# АДМИНСКИЕ ШТУЧКИ

Если все совсем плохо

**strace**
**netstat**

**ps**
**htop**

**tcpdump**
**и т.п.**

# ДЕЙСТВИЯ ПРИ ЧП!

- НЕ ПАНИКОВАТЬ, НЕ ВЫКАТЫВАТЬ ХАОТИЧЕСКИЕ «ХОТФИКСЫ»
- СОБРАТЬ КАК МОЖНО БОЛЬШЕ ИНФЫ
- НЕ СТЕСНЯТЬСЯ СПРАШИВАТЬ ИДЕЙ У КОЛЛЕГ
- ГУГЛИТЬ
- ИСКАТЬ И ИСПРАВЛЯТЬ ПРИЧИНУ, А НЕ СЛЕДСТВИЕ

# ПРИМЕР ИЗ ЖИЗНИ

# ПРОБЛЕМЫ С ОДНИМ ГЕО-СЕРВИСОМ

Проекту около 5 лет

Нагрузка средняя

Есть внешние API

# ПРОБЛЕМЫ С ОДНИМ ГЕО-СЕРВИСОМ


Ruby 2.1


Rails 4.1


Unicorn


PostgreSQL


PostGIS

Resque заменили на Sidekiq — начались проблемы

# ЧТО СЛУЧИЛОСЬ?

RGeo::Error::ParseError: Not enough bytes left to fulfill 1 byte

Для пользователя некоторые запросы не обрабатывались, частично функционал не был досутпен.

# АНАЛИЗ СИТУАЦИИ

Код не менялся

Стектрейс ведет в lib/coords.rb

```
@@parser = RGeo::WKRep::WKBParser.new
def coords
 @@parser.parse(geo_coords)
end
```

# РЕЗУЛЬТАТ АНАЛИЗА: ОДИН WKBPARSER НА ВСЕ ПОТОКИ

WKBParser не является потокобезопасным

Активно используются instance-переменные

# ЕЩЕ ПРИМЕР

# ЧТО СЛУЧИЛОСЬ?

502 на все запросы

# АНАЛИЗ СИТУАЦИИ

Процессы работают

CPU — ОК

RAM — ОК

Сеть — ОК

В логах пусто — кого-то ждем

# ПРОБЛЕМА 1

Причина: strace — подвисли на запросе к API
без API можно обработать 85% запросов

Решение: мониторим состояние API и если все плохо — просто не делаем запросов

# ЕЩЕ ПРИМЕР

# ЧТО СЛУЧИЛОСЬ?

502 на все запросы
Для пользователя тоже самое

# АНАЛИЗ СИТУАЦИИ

В логах — началась обработка запроса и висит

# ПРОБЛЕМА 2

Причина: Длинные транзакции в sidekiq воркерах
Превышение лимита одновременных транзакций

Решение: Делаем в транзакции только самое необходимое, остальное — вне транзакции.

# ЕЩЕ ПРИМЕР

# ЧТО СЛУЧИЛОСЬ?

502 на все запросы

# АНАЛИЗ СИТУАЦИИ

CPU — ОК

RAM — ОК

Сеть — ОК

В логах — пусто. Воспроизводится не регулярно

# АДМИНЫ БД ГОВОРЯТ — ПРОБЛЕМА 2

# ТВОЙ СОФТ — ГОВНО!

# АНАЛИЗ СИТУАЦИИ

```
def fill_location(loc)
 if loc
   self.coords_source = loc.source
   self.coords = loc.coords
   fill_address_if_required
 end
 self.state = 'processed'
 save! # ← зависает тут
end
```

# АНАЛИЗ СИТУАЦИИ

```
BEGIN TRANSACTION;
UPDATE ...;
COMMIT;
```

# Я ТЕБЕ НЕ ВЕРЮ!

PFF...

# ПРОБЛЕМА 3

pg 0.14.1 (2012 год)

# ПРОБЛЕМА 3

Зависание — в нативном расширении.
Обновление гема pg спасло мир!

# ВЫВОД

Отладка - набор эвристик + метод научного тыка.
Чем лучше эвристики, тем выше вероятность того, что проблема будет решена быстро.

# ВОПРОСЫ?