Кое-что про Erlang

а также OCaml, Haskell Perl, PHP, C и C++

Лев Валкин, 2010

О докладчике

- Драйверы на Asm x86 (1993)
- Веб-сайты на C, Shell, Perl (1996+)
- Математические обучалки на Delphi (1998)
- SMS-гейт на C, C++, Visual Basic (2001)
- Ускорял HTTP[S] на C, M4 (2003, два раза изобрёл LISP)
- Куски embedded-платформы Cisco ASA/PIX

Стар(та)пер-2006

- AboutEcho.com
- Анонсы на TechCrunch, RRW, VentureBeat
- ~$4.8mln венчурного финансирования
- ~25 человек, из них 20 — в России

2006: Быстрый старт

- Цель: быстро сделать эксперимент
- 4 дня программирования на Perl, через неделю — анонс на TechCrunch (2006)
- Привалило трафика (Sun SPARC 600 MHz)
- Часть функциональности сброшено на C (для скорости)

2007: Развитие

- Количество программистов: 1
- Perl; в критических участках — C
- Сложность дебага: 0
- Сложность развития: удовлетворительная
- Когда нет команды, любая знакомая технология будет приемлемо работать!

2007: Развитие

- Количество программистов: **4**
- Perl; в критических участках — C
- Сложность дебага: **начинаются проблемы**
- Развитие:
  - Perl: разводится бардак в коде
  - C: разводятся sigsegv в продакшне

2008: Развитие

- Количество программистов: **5-10**
- В Perl растёт лапшизм в коде
- Perl заменяется на Erlang (для нового кода)
- Те же люди (!)
- Без опыта в ФП (!!)
- В условиях ограниченного времени и бюджета

Интермиссия: Erlang

- Считается функциональным языком
- Неизменяемость «переменных»
- Горячая замена кода
- «Конкурентно-ориентированный»
  — Процессы изолированы, общаются сообщениями
- Чрезвычайно простой

Простой Erlang

- «Erlang действительно учится за две недели, это не миф»

  http://grey-kristy.livejournal.com/87271.html

- «По информации от различных источников: 2 недель хватит»

  http://levgem.livejournal.com/285670.html

Мини-опрос

- Спросил наших разработчиков
- «Я бы положил на эрланг от одной до двух недель плотного изучения с упражнениями».
- «Достаточно одной — двух недель (прочитать getting started guide или книгу Армстронга)».
- «Можно учить специально и непрерывно, а можно по наличию задач. Изучал по второму варианту; за месяц».

Мини-опрос

- «Писать можно на эрланге через неделю — с докой, через две — с докой на полке, через месяц — комфортно».
- «Уверен, что можно выучить erlang и уже попробовать OTP за месяц. Без OTP вообще наверное недели две».
- «По эрлангу, чтобы начать писать модули — недели две чтения мануалов и тьюториалов. И еще пару недель доучиваться erlang-style кодописанию».

С точки зрения руководителя

- Нанимаемые специалисты имели опыт от PHP +JavaScript до C/C++, но не имели опыта в ФП.
- Предыдущий опыт на скорость обучения не влияет.
- Возраст — 22-35 лет, на скорость изучения не влияет.
- При наличии задач, обучение происходит быстро, в горизонте оперативного планирования (2-4 недели).

С точки зрения code nazi

- «Набеговое программирование» в команде располагает делать спагетти-код, кишащий сайд-эффектами.
- Erlang «сопротивляется» эффектам, к которым приводит спагетти-код:
- Процессы изолированы, эффекты плохого кода в одной подсистеме не влияют на другую.
- Связанные процессы не позволяют редкой ошибке привести к отказу системы.

С точки зрения code nazi

- Неизменяемые переменные и отсутствие глобальных переменных поощряют разработку с минимумом сайд-эффектов.
- Эти же свойства упрощают независимое тестирование подсистем.

Адвокат дьявола

- В распространённых языках проблема спагетти-кода облегчается инструктажем, соглашениями о коде, юнит-тестами и ревью кода. А если проблема не стоит, зачем использовать Erlang? Тем более, для программ в функциональном стиле сложно проводить code-review.

- В нашей компании делается ревью Erlang-кода. Спросим инженеров, что легче, ревью скриптового языка, на котором они писали до эрланга, или ревью Erlang-кода?

Ревью P* vs. Erlang?

- *(Знание языков. «Отзыв»)*
- Erlang >> P*. «Конечно, [Erlang] проще. Меньше возможностей для сайд-эффектов. Функции из других модулей qualified, проще искать код».
- Erlang (0.7) Perl (0.5). «[Erlang] Проще».
- Erlang (0.8) Perl (0.5). «[Erlang] Однозначно проще. Когда пару раз столкнешься с проблемами в перле, начинаешь понимать, что ревью там делать очень сложно».

Ревью P* vs. Erlang?

- PHP (1.0) Erlang (0.7). «Одинаково».
- PHP (0.7) Erlang (0.6). «Владея контекстом, легче делать ревью на erlang код».
- Perl (0.9) Erlang (0.8). «Думаю, одинаково. Ревью функционального кода требует более глубокого вникания, но менее "богатый" синтаксис и компактный код упрощают понимание».
- Perl (1.0) Erlang (0.8). «Проще [Erlang]».

Ревью P* vs. Erlang?

- Те, кто знают Erlang лучше (кстати, почему?), говорят, что для Erlang проще делать код-ревью. Логично.
- Те, кто знает P* языки лучше, говорят, что для Erlang не сложнее делать код-ревью. Почему?
- «Менее богатый синтаксис и компактный код упрощают понимание».
- «Меньше возможностей для сайд-эффектов».

2009: Ускорение

- Напомню: Perl → (Perl, C) → (Erlang, C/C++)
- Проблема: никто не хочет работать с C/C++. Сложно найти или обучить людей (N.B.: обычно этот аргумент выдвигают против FP!)
- «Набеговое программирование» даёт сбой и здесь. Sigsegv в продакшн становятся ежедневной проблемой, а то и бизнес-риском. Valgrind и анализ корок помогают, но симптоматично, не устраняя причины.

2009: Ускорение

- Идея: **попробовать Haskell** (я знал неплохо).
- Риски: **кривая обучения**, производительность решения.
- Сделан аналог C++ кода на Haskell.
- В 10 раз медленнее (в 10 раз больше машин?!)
- После оптимизаций — в 3 раза медленнее. Плохо.

2009: Ускорение

- Альтернатива: попробовать OCaml (никто не знал).
- Риски те же: кривая обучения, производительность.
- Аналог нашего основного C++ компонента:
  - Решение в лоб: ~5-10% быстрее (да, свежий код).
  - В ~5 раз — на несколько тысяч строк — короче (см. рис.)
  - Пришлось задействовать **две** глобальные переменные.

Явные аннотации типов

```
std::map<std::string,EntityTypeID>::iterator found =
```

2010: Настоящее

- Эволюция ядра системы:

  Perl → (Perl, C) → (Erlang, C/C++) → (Erlang, OCaml)

- Другие языки тоже используются: Perl, PHP, Python, JavaScript (на клиенте), C, C++, Haskell (для генерации JS).

- Примерно в 3-5 раз больше функционального кода.

Кококомл

- — Он спрашивает, «как ока?»
  — Ну и ответь ему, «ко-ко-ко»

- Изучаемость: «Примерно в два раза дольше, чем эрланг». «Без опыта в ФП — месяца два, если есть опыт, то хватит месяца». (А также «две недели» и «месяца три»)

- «Окамл я знаю хуже C++, но разбираться в нем (окамле) проще, чем в C++». Кстати, как в OCaml с код-ревью?

Ревью C/C++ vs. Ocaml

- *(Знание языков. «Отзыв»)*
- C+/C++ >> OCaml: «По сравнению с c/c++ окамл я знаю намного хуже. При солидной форе в сторону c/c++ я считаю, что код-ревью в ocaml проще, опять же, из-за почти отсутствующих сайд-эффектов».
- OCaml (0.4): «Сложнее».
- C/C++ (0.8) OCaml (0.6): «Для меня пока сложнее или одинаково».

Ревью C/C++ vs. Ocaml

- OCaml (0.7) C (0.5): «Сложнее. Причина в основном в каринге и использовании функций без точечной нотации».
- OCaml (0.6) C (0.6): «Если изменения в коде масштабные, ocaml ревьюить сложнее».

Erlang vs. OCaml executive summary

- Erlang быстро изучается и положительно воспринимается командой, быстро переходя в список «комфортных, своих» языков.

- «Из всех перечисленных языков эрланг нравится больше всех, не для решения конкретных задач, а просто по ощущениям» (респондент знает PHP глубже, чем Erlang)

- OCaml воспринимается тяжеловато, как типичный «неосновной язык». Делать ревью кода для него сложнее, чем для C/C++. Может ли это быть артефактом лаконичности (количество смысла на строчку) кода?

Erlang vs. OCaml executive summary

- Инженеры отметили положительное влияние отсутсвия сайд-эффектов на программирование и код-ревью OCaml и Erlang.
- Erlang прост! Необычность синтаксиса более чем компенсируется его простотой. Понятная и элементарная семантика. REPL.
- Идиосинкразии (не просто необычности, а реальный shit) синтаксиса OCaml вызывают нарекания в процессе обучения. Ограниченный REPL.
- OCaml компилируем и быстр! Если не делать специальных оптимизаций, то OCaml ~= C/C++, особенно если переписывать существующий код.

Erlang

- Erlang хорошо себя зарекомендовал для общего серверного программирования: command-and-control, координирование потоков данных, управление функциональностью, написанной на других языках (OCaml, C/C++, Perl, Python).
- В нашей компании он занял нишу динамических серверных языков (Perl, PHP, Python).
- Независимость процессов и лёгкость коммуникации между ними приводят к простоте решений и изоляции негативных эффектов. В продакшн можно пускать неидеальный код.

OCaml

- OCaml хорошо зарекомендовал себя при необходимости делать символические вычисления, порождая эффективный код.
- В нашей компании он занял нишу C/C++.
- Если есть возможность, рекомендую учить Haskell (ради стройности на концептуальном и синтаксическом уровне), затем использовать OCaml, если появится нужда в скорости, предсказуемости поведения или желание иногда использовать сайд-эффекты без излишнего монадизма.

Вопросы?

xkcd.com/297