Опыт работы с LuaJIT в нагруженных интернет-проектах

Александр ГЛАДЫШ LogicEditor, CTO ag@logiceditor.com

RIT++ 2013 r

- 1. Почему Lua?
- 2. O Lua и LuaJIT
- 3. Почему не писать всё РНР?
- 4. О существующих решениях для реализации веб-сервисов на Lua
- 5. Наш нынешний стек
- 6. Грабли
- 7. Каким мы видим стек следующего поколения?
- 8. Хотите знать больше?
- 9. Вопросы?



Мы вышли из игровой индустрии, где Lua правит миром.

Прагматически Почему Lua?

- ▶ Работает быстро!
- ▶ Писать удобно!
- ▶ Освоить легко!

Недостатки Почему Lua?

- В первую очередь где искать людей?
- ▶ Основные проблемы при переучивании на Lua.
- Идеосинкразии языка.
- ▶ Пишите на Lua как на Lua!

Очень кратко о языке Lua, его происхождении, особенностях и росте популярности в последние годы. Где используется язык? IDE и специализированные IDE. Мэйнстрим и самопальные диалекты, NIH-синдром и лёгкость доработки напильником. Цена и выгоды отхода от мэйнстрима. Lua 5.1 vs. Lua 5.2. Metalua.

LuaJIT 2.0: почти-мэйнстрим диалект Lua. JIT, FFI, производительность. Поддерживаемые платформы. Ограничения на 64-х битах. LuaJIT vs. Lua 5.2. Вкусности, планируемые для LuaJIT 2.1 и LuaJIT 3.

Встроенный vs. расширяемый язык (на самом деле и то и то)? Ситуация до LJ2 и после. Теперь можно больше не писать на C!

Раньше с кодом было туго, сейчас качественного готового кода на Lua много. LuaRocks.

Место для Lua / LuaJIT в вашем стэке? Как другие интернет-системы используют Lua? Как это делаем мы? С нами "всё ясно", мы — хардкорщики из геймдева (на самом деле нет). Почему и где стоит начать применять технологии из этого доклада в существующих продакшен-системах?

а) Настраиваемая пользователем логика.

b) Отдельностоящие сервисы.

с) Код, который иначе был бы написан на C/C++/OCaml.

d) ...

О существующих решениях для реализации веб-сервисов на Lua

Популярные: a) Kepler/WSAPI — дёшево и сердито. b) Luvit — модная бяка, навязывает чуждую мэйнстримному Lua нодовскую экосистему. c) openresty — перспективный продукт китайской инженерной мысли.

Остальные — см. ТООО

а) Какие задачи мы решаем?

b) На каком железе мы живём?

c) Архитектура взаимодействия. XEN, Ubuntu (и её тюнинг), nginx (и его тюнинг), spawn-fcgi, multiwatch, LuaJIT 2, WSAPI, 0MQ, Redis (и его тюнинг). DNS-ы. Отдельностоящие сервисы. Почему так?

d) Какие луашные библиотеки мы используем и почему? Годные альтернативы нашим историческим opensource-велосипедам (и какие из велосипедов — лучше альтернатив).

e) Как сделано High Availability?

f) Как устроен деплоймент?

g) Как устроен мониторинг?

h) Какие показатели по производительности? По стабильности?

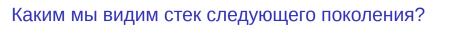
i) DSL для описания обработчиков запросов. Кодогенерация. Прочие рюшечки и сахар (бонус: DSL для описания SQL-данных с возможностью автогенерации продвинутого UI бэкофиса для этих данных).

а) Какие были основные проблемы? Как их решали? Несколько общих советов по отладке и оптимизации производительности при работе с Lua. Отладка отладчиком и по логам, оптимизация GC, какие параметры нужно мониторить. Профайлинг кода на LJ2. Автотесты.

b) Какие проблемы не решены, и как c этим жить?

i. Long polling / comet.

ii. TODO



а) Ориентироваться на openresty, но не использовать его напрямую. Почему?

b) Новая архитектура.

і. Проще! Ещё проще!

іі. Отказ от LuaRocks.

ііі. Полный переход на FFI.



iv. Отказ от FCGI и WSAPI. Переход на epoll и библиотеку парсинга HTTP.

v. Отказ от сервера конфигураций.

vi Улучшенная High Availability.

vii Неблокирующее API на корутинах, без коллбэков. Архитектура. Особенности реализации для основных сервисов (HTTP[S], Redis, MySQL/Postgres).

viii. Новый дизайн DSL.

ix. ...

Хотите знать больше?

9. Рекомендуемые источники информации о Lua, LuaJIT и сопутствующих технологиях.

Вопросы?

ag@logiceditor.com