# Erlang and Thrift for Web Development

Todd Lipcon
(@tlipcon)

Cloudera

March 25, 2010

cloudera

Introduction

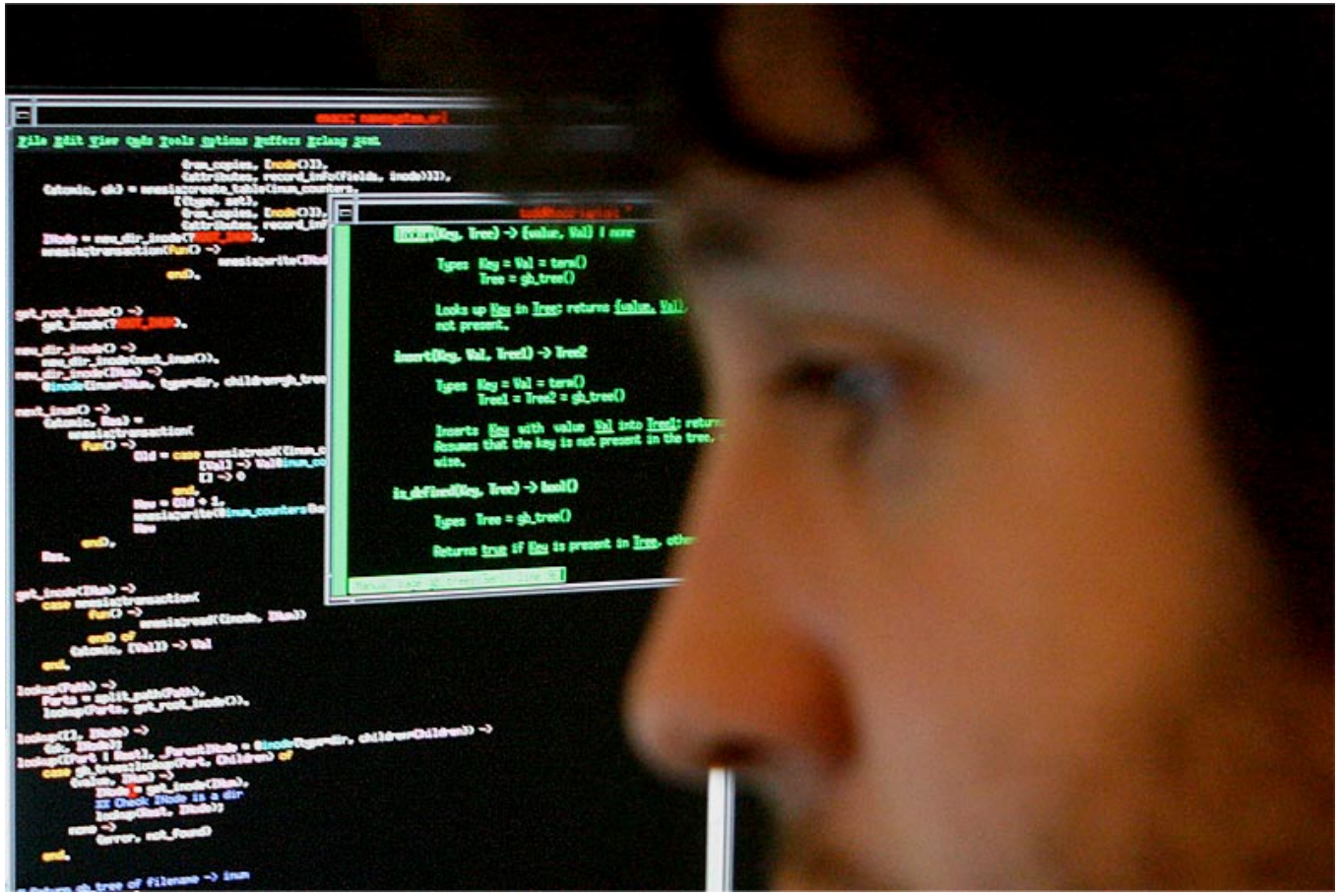
Erlang vs PHP

Thrift

A Case Study

cloudera

# About Me




cloudify looks like @**cloudera** is doing some #erlang too, I wonder if they integrate it with #hadoop http://tinyurl.com/y8kvtf3

1 day ago from web

dera

# About Me

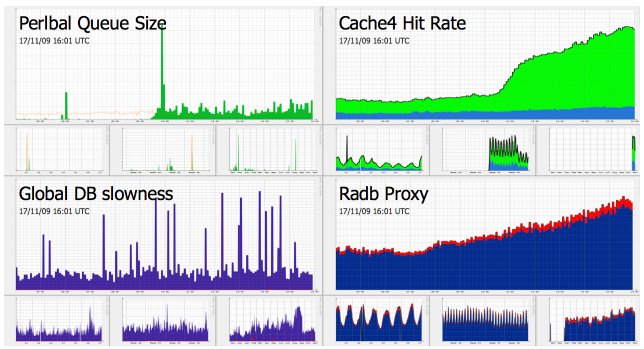
- Built many web sites in Perl, Ruby, Python, PHP, Java, and Erlang
- Previously at AmieStreet.com / Songza.com
  - Erlang, PHP, Python, Java
  - (Re)wrote Erlang Thrift bindings
- Apache Thrift committer
- Now at Cloudera, working on Hadoop and HBase (unrelated, but ask me about it!)

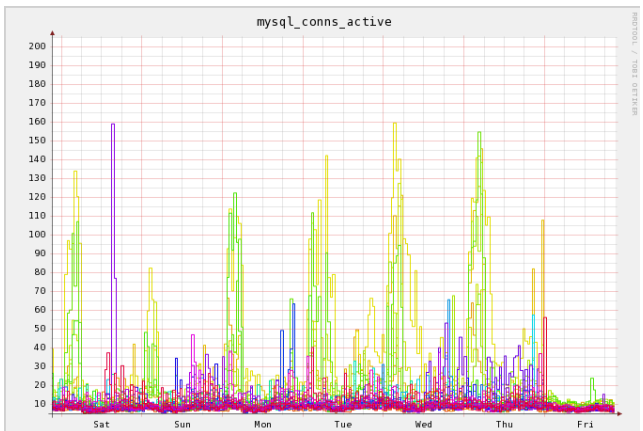
cloudera

# Scope

## You might care about this talk if your web site is...

- mostly dynamic content
- built by multiperson/multiskill teams
- hosted on dedicated machines
- your fulltime job
- trying to do something complicated




cloudera

# Popular Web Languages

...until next year

- PHP
- Ruby
- Python
- Scala? Clojure?
- ASP.NET?

# Popular Web Languages

...until next year

- PHP
- Ruby
- Python
- Scala? Clojure?
- ASP.NET?

What have they got that make them excel for web development?

cloudera

# Where PHP Excels

This page intentionally left blank.

# Where PHP Excels

...seriously!

- No concurrency model
- Templating, string manipulation
- Implicit casting, “sloppy” semantics
- Availability of web frameworks, common code, etc
- Availability of designer-programmers
- Large existing codebases

# Where Erlang Excels

## Preaching to the Choir

- Great concurrency model
- Great reliability features
- Achieving 5 nines is relatively easy
- Dealing with inter-process communication and async processes is natural

# Where PHP sucks

## for the forseeable future

- No concurrency possible.
- All inter-request communication must go through an external party
- Each thread ties up a web server process
- Asynchronous actions are hard
- Ever seen a daemon in PHP?

# Where PHP sucks

## for the forseeable future

- No concurrency possible.
- All inter-request communication must go through an external party
- Each thread ties up a web server process
- Asynchronous actions are hard
- Ever seen a daemon in PHP?
- Did you still see it 100M requests later?

# Where Erlang sucks

...at least, for now

- Template designers can't pick it up easily ("weird syntax")
- Immutability feels unnatural
- String manipulation, unicode support, etc
- Obtuse error printouts
- Fewer web frameworks

# An observation

- Where PHP sucks is where Erlang excels!
- And vice versa!

# An observation

- Where PHP sucks is where Erlang excels!
- And vice versa!
- Wouldn't it be nice to have the good parts of both?

# An observation

- Where PHP sucks is where Erlang excels!
- And vice versa!
- Wouldn't it be nice to have the good parts of both?
- Let's glue them together!


cloudera

# Enter Thrift

...mmmm... glue...

- Thrift is glue that makes multilingual development easy!
- Cross-language RPC and serialization library
- Bindings for C++, C#, Java, Python, Ruby, Perl, PHP...
- plus Haskell, Smalltalk, ObjC/Cocoa, OCaml

# Enter Thrift

...mmmm... glue...

- Thrift is glue that makes multilingual development easy!
- Cross-language RPC and serialization library
- Bindings for C++, C#, Java, Python, Ruby, Perl, PHP...
- plus Haskell, Smalltalk, ObjC/Cocoa, OCaml
- And of course: Erlang!

cloudera

# A Touch of History

- Originally developed by Facebook (mainly PHP shop)
- Open sourced in Spring 2007
- Now in Apache Incubator, 0.2.0 released in Dec '09
- Reasonably widespread usage



cloudera

# Thrift Features

## Serialization

- Primitives and complex datatypes
- Cross-platform cross-language
- Multiple *Protocol* implementations
- Backwards compatibility built in
- Useful for long-term storage, too

# Thrift Features

## RPC

- Makes remote interlanguage function calls feel like local ones
- Serializes calls, results, exceptions over a *Transport* (eg socket)
- Provides *Service* and *Client* abstractions
- Comes with well-written client and server skeletons

# Why Design with Services?

...promise this is the only slide with "SOA" on it

A service-oriented-architecture gives you:

- Modularity with clean APIs
- Vertical partitioning for scalability, hardware specialization, or *language* specialization
- Long-lived data in RAM

e.g: Search, Storage, "Smart Data"

cloudera

# Thrift vs other options

- CORBA - less language support, totally unfriendly
- Protobuffers - OSS version doesn't include RPC stack
- Roll-your-own - bug prone and tedious
  - Though marginally more efficient
- HTTP/REST/JSON - deep structures without types are inconvenient

# Steps to use Thrift

1. Write a `.thrift` file

# Steps to use Thrift

1. Write a `.thrift` file
2. Run `thrift -gen erl -gen py foo.thrift`

# Steps to use Thrift

1. Write a `.thrift` file
2. Run `thrift -gen erl -gen py foo.thrift`
3. Do some real work (fill in implementation)

# Steps to use Thrift

1. Write a `.thrift` file
2. Run `thrift -gen erl -gen py foo.thrift`
3. Do some real work (fill in implementation)
4. Profit

# Sounds like fun!
# DEMO!

# A Case Study

## Amie Street Pricing Server

- AS's first project in Erlang
- Handles all dynamic prices and commerce transactions
- Runs on a non-dedicated pair of nodes


cloudera

# Dynamic Pricing on Amie Street

- Online mp3 store with dynamic pricing
- Each time a song is bought, its price increases
- Prices are functions of the number of previously completed buys
- Can never sell cheaper than the price function

# Dynamic Pricing on Amie Street

- Online mp3 store with dynamic pricing
- Each time a song is bought, its price increases
- Prices are functions of the number of previously completed buys
- Can never sell cheaper than the price function
- This is actually really tricky!

# What to do about concurrency?

- Alice goes to AmieStreet.com and sees a song at 30 cents.
- Bob also sees the same song at 30 cents.
- They both click "buy" at the same time, and see a confirmation dialog for their item at 30c.
- Alice confirms payment and receives song at 30 cents.
- What price does Bob get?

# The Solution

- Give everyone *tickets* at price points
- Expire those tickets for non-conversions, logouts, etc
- Sounds like a problem for Erlang!
- Model carts as processes, linked to `ticket_releasers` which handle cleanup, etc.

# The Solution

- Give everyone *tickets* at price points
- Expire those tickets for non-conversions, logouts, etc
- Sounds like a problem for Erlang!
- Model carts as processes, linked to `ticket_releasers` which handle cleanup, etc.

No idea how we would have solved this in PHP

cloudera

```
GetCartResult getCart(
  1:ReqInfo info,
  2:list<RequestedCartObject> requested_objects)

BuyCartResult buyCart(1:i32 user_id, 2:i32 uniq_id)

bool cancelCart(1:i32 user_id, 2:i32 uniq_id)

list<PriceInfo> getAlbumPriceInfo(
  1:ReqInfo info, 2:list<i32> album_ids)

list<PriceInfo> getSongPriceInfo(
  1:ReqInfo info, 2:list<i32> song_ids)
```

# Results

- We shipped in about a month and a half
- $\sim$4000loc[1], with lots of new features
- Separated the difficult distributed system from the PHP code
- Black box "in a good way" to front-end engineers
- Very stable and performant!

[1] as of 1/09

cloudera

# More Case Studies

## Facebook Chat

- MochiWeb "channel" servers for long poll
- Uses `thrift_client` to talk to presence servers (C++)
- Uses server to hear events from PHP
- Read the FB Eng blog for detailed info and a neat video

cloudera

# More Case Studies

songza listen. now.
the music search engine & internet jukebox

- "Web jukebox" aggregates media searches from several backend APIs
- Used to be serial curl requests from PHP
- Moved to an Erlang Thrift service to do requests in parallel
- Way easier! Took 2-3 days for an Erlang n00b

cloudera

# What's up next?

(the new hotness)


- More compact serialization
- More oriented towards large dataset storage
- Erlang bindings in progress![2]

[2] help wanted, contact me!

cloudera

# Links

- Thrift: `http://bit.ly/thrift`
- ThriftErlSkel: `http://bit.ly/terlskel`
- Twitter - `@tlipcon`

cloudera

# Backup Slides

# A .thrift file

```
exception MathException {
  1:string message
}

service CalculatorService {
  i32 add(1:i32 a, 2:i32 b)
  i32 subtract(1:i32 a, 2:i32 b)
  double divide(1:double a, 2:double b)
    throws (1:MathException me)
}
```

# Thrift Data Types

...basically what you expect

- `bool, byte, i16, i32, i64, double`
- `string, binary`
- `list<T>, map<K,V>, set<T>`
- `struct T { ... }, exception X { ... }`