Erlang *(and Python)*

Lawrence Oluyede

why do we bother?

(thanks David!)

It's about **concurrency**. It's about **distribution**. It's about **fault tolerance**. It's about **functional programming**. [...] It's about [...] a process I call **concurrency-oriented programming**.

Joe Armstrong, Programming Erlang

Why not Erlang?

- Odd syntax
- Variables are immutable once given a value
- Functional programming

Why Erlang?

- Odd syntax
- Variables are immutable once given a value
- Functional programming

How does it look like? (1)

```
Erlang (BEAM) emulator version 5.6.5 [source] [smp:2] [async-
threads:0] [hipe] [kernel-poll:false]
Eshell V5.6.5  (abort with ^G)
1> 1.
1
2> 123456789 * 987654321.
121932631112635269
3> hello.
hello
4> X = 1234.
1234
5> X = a.
** exception error: no match of right hand side value a
```

How does it look like? (2)

```
1> Me = {person, {name, lawrence}, {age, 25}}.
{person,{name,lawrence},{age,25}}
2> Me.
{person,{name,lawrence},{age,25}}
3> {person, {name, Name}, {age, Age}} = Me.
{person,{name,lawrence},{age,25}}
4> Name.
lawrence
5> Age.
25
```

How does it look like? (3)

```
1> "Lawrence".
"Lawrence"
2> [76, 97, 119, 114, 101, 110, 99, 101].
"Lawrence"
3> [1, 76, 97, 119, 114, 101, 110, 99, 101].
[1,76,97,119,114,101,110,99,101]
```

How does it look like? (4)

```
6> [this, is, a, list, with, Me].
[this,is,a,list,with,{person,{name,lawrence},{age,25}}]
7> [1, 2, 3, 4, 5].
8> [H|T] = [1, 2, 3, 4, 5].
[1,2,3,4,5]
9> H.
1
10> T.
[2,3,4,5]
11> [2 * X || X <- T].
12> lists:map(fun(X) -> 2 * X end, T).
[4,6,8,10]
```

```
>>> T = [2, 3, 4, 5]
>>> [2 * X for X in T]
[4, 6, 8, 10]
```

How does it look like? (5)

```
-module(temp).
-export([convert/1]).

f2c(F) ->
    ((F - 32) * 5) / 9.

c2f(C) ->
    (C * 9) / 5 + 32.

convert({c, N}) ->
    {f, c2f(N)};

convert({f, N}) ->
    {c, f2c(N)}.
```

Concurrency

- Lightweight processes sharing **nothing**
- Primitives are basically spawn(), !, receive
- Erlang processes are not OS processes
- Processes are functions executed concurrently

Concurrency example (part 1)

```
-module(pingpong).
-export([start/1, ping/2, pong/0]).

pong() ->
    receive
        {ping, PingPid} ->
            io:format("Pong got ping~n", []),
            PingPid ! pong;
        die ->
            io:format("Pong HAS DIED!~n", []),
            exit(bye)
    end,
    pong().
```

Concurrency example (part 2)

```
ping(0, PongPid) ->
    PongPid ! die,
    io:format("PING HAS DIED!~n", []);

ping(N, PongPid) ->
    PongPid ! {ping, self()},
    receive
        pong -> io:format("Ping got pong~n", [])
    end,
    ping(N-1, PongPid).


start(N) ->
    PongPid = spawn(?MODULE, pong, []),
    spawn(?MODULE, ping, [N, PongPid]).
```

Concurrency example (output)

```
$ erl
1> c(pingpong).
{ok,pingpong}
2> pingpong:start(3).
Pong got ping
<0.39.0>
Ping got pong
Pong got ping
Ping got pong
Pong got ping
Ping got pong
PING HAS DIED!
Pong HAS DIED!
3>
```

Erlang <=> Python

- Linked-in drivers
- Ports
- Sockets
- HTTP
- Disco

Linked-in drivers

- External program linked as as shared library
- Unsafe
- Lowest level possible

Ports

- Two way byte-stream channel between the Erlang VM and an outside process
  - open_port(PortName, PortSettings) -> port()
  - Safe

Sockets (server)

```
handle(Socket) ->
    receive
        {tcp, Socket, Bin} ->
            Received = binary_to_list(Bin),
            io:format("Received: ~p~n", [Received]),
            Response = "Hello, you!",
            gen_tcp:send(Socket, Response),
            handle(Socket);
        {tcp_closed, Socket} ->
            io:format("Socket closed~n")
end.
```

Sockets (client)

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('localhost', 9999))
text = 'Hello, server!'
tosend = [struct.pack('!h', len(text)), text]
for msg in tosend:
    s.send(msg)
print "Client received: %s" % s.recv(1024)[2:] # discard length
s.close()
```

HTTP

- Use MochiWeb for the server (http://code.google.com/p/mochiweb/)
- Easy to write JSON based web services
- urllib as the client + (simple)json

Disco

- Map Reduce implementation for clusters
- Core in Erlang and everything else in Python
- Jobs are written in Python (pure) functions
- Used for full text indexing, data analysis, log parsing
- Can be used on Amazon EC2

Who uses Erlang?

facebook

RabbitMQ
Open Source Enterprise Messaging

apache CouchDB relax

ejabberd
the Erlang Jabber/XMPP daemon
community site

mochimedia

adroll
get targeted

tranquo

delicious

"When in doubt create a new process"

-- *Gabriele Lana*

Who am I

Lawrence Oluyede, Developer @
StatPro Italy

lawrence.oluyede@statpro.com
l.oluyede@gmail.com

http://twitter.com/lawrenceoluyede

http://www.linkedin.com/in/
lawrenceoluyede