

Why Functional Programming?

An attempt to justify Erlang's existence
Martin Owen

```
1 partition(F, L) ->
2   partition(F, L, [], []).
3
4 partition(F, [HIT], Yes, No) ->
5   case F(H) of
6     true ->
7       partition(F, T, [HIYes], No);
8     false ->
9       partition(F, T, Yes, [HINo])
10  end;
11 partition(_, [], Yes, No) ->
12  {Yes, No}.
```

A Functional Function

```
lib_misc:partition(  
  fun(Num) -> Num >=5 end,  
  [3, 8, 5, 10, 4, 2, 6]).
```

```
{[6,10,5,8],[2,4,3]}
```

Result

Why?

- Increases in processor clock speeds are leveling off.
- Multi-core CPUs are increasing.
- This is likely to mean a paradigm shift in software development.
- Erlang may not be the language to lead this shift, but the concepts that it introduces will be important.

Why Erlang? In short, because the problem domain fits Erlang like a glove. Erlang is a functional concurrency-oriented language with extremely low-weight user-space "processes", share-nothing message-passing semantics, built-in distribution, and a "crash and recover" philosophy proven by two decades of deployment on large soft-realtime production systems.

Eugene Letuchy - Facebook

[http://www.facebook.com/notes.php?
id=9445547199](http://www.facebook.com/notes.php?id=9445547199)

Erlang/Open Telecom Platform

- Developed by Ericsson 20 years ago, released as open source 10 years later.
- Not just a language - a **platform** for developing fault-tolerant, scalable, distributed applications.
- Includes frameworks for writing your own servers (with features such as hot code swapping) and a fast DBMS (Mnesia).

Eshell V5.6 (abort with ^G)

1> Num = 5.

5

2> Num = 12.

* * exception error: no
match of right hand side
value 12

3>

Immutable Code

Erlang has no mutable
data structures

No mutable data
structures = No locks

No mutable data
structures = Easy to parallelize

Concurrency-Oriented Programming

What's going to really fuck people up, out in the real world, is the programming model. Programming is hard; parallel programming is way the hell harder; compsci courses have turned into votech Java pap; and enrollments in compsci are in any case as lively as the waiting list for the Lusitania the week after it was torpedoed. People want their programming to be easier and more casual, and they're about to have it jammed into their eyesockets on bamboo stakes instead.

Bryan O'Sullivan

Author of Real World Haskell

Processes

- Concurrency works best when there are lots of processes.
- In Erlang, processes belong to the language **not** the operating system.
- Spawning processes, and inter-process communication is built into the language.

Processes In Erlang

- Creating and destroying them is very fast.
- Sending messages between them is very fast.
- They behave the same way on all operating systems.
- We can have very large numbers of them.
- They share no memory and are completely independent.
- The only way they interact is through message passing.

Concurrency Primitives

- Spawn a new process
 - `Pid = spawn(Fun)`
- Send a process a message
 - `Pid ! Message`
- Receive a message from a process
 - `receive ... end`

```

1 pmap(F, L) ->
2   S = self(),
3   %% make_ref() returns a unique reference to match
4   Ref = erlang:make_ref(),
5   Pids = map(fun(I) ->
6     spawn(fun() -> do_f(S, Ref, F, I) end)
7     end, L),
8   %% gather the results
9   gather(Pids, Ref).
10
11 do_f(Parent, Ref, F, I) ->
12   Parent ! {self(), Ref, (catch F(I))}.
13
14 gather([Pid|T], Ref) ->
15   receive
16     {Pid, Ref, Ret} -> [Ret|gather(T, Ref)]
17   end;
18 gather([], _) ->
19   [].

```

A Parallel Function

Distributed programs

- Can pass messages over a network to other Erlang “nodes”.
- Spawn a process on another node
 - `Pid = spawn(Node, Mod, Fun, ArgList)`
- Distributed Erlang code doesn't look much different to regular code.

Error Primitives

- Spawn a new process and link to the parent
 - `Pid = spawn_link(Fun)`
- The parent will be notified of any errors
- Processes can monitor each other

What Erlang is good at

- Servers.
- Manipulating binaries.
- Code that needs to scale quickly.
- Fault-tolerant code.

The Negatives

- Strings are lists of Integers.
 - "Martin" ::= [77,97,114,116,105,110].
- Language feels more “C-like”.
- Closest thing to an IDE is GNU Emacs.
- What Sucks About Erlang
 - http://damienkatz.net/2008/03/what_sucks_abou.html

Further Resources

- Programming Erlang by Joe Armstrong
- Erlang Homepage
 - <http://www.erlang.org>
- Google Tech Talk on Erlang
 - <http://www.youtube.com/watch?v=OpYPKBBQhSZ4>
- Erlang - The CEOs Point of View
 - <http://oreillygmt.typepad.com/scenius/2008/02/erlang-the-ceos.html>
- Planet Erlang (Digg-style news site)
 - <http://www.planeterlang.org>