HiPE & Erlang internals

at

London Erlang User Group meeting
2010-05-26

by

Daniel Luna

Daniel Luna

Spends his time with Erlang hacking, Junior Chamber International (JCI), the board game Go, travel, financial systems, personal development, making new friends, and his girlfriend.

Based in South Kensington, London

HiPE, Uppsala University, 2003-2005

Master's thesis, published at EW '04 & PPDP '05

Ericsson, Rome, 2006

Consultant for Erlang Solutions

Klarna (Kreditor), Stockholm, 2006-2009

Erlang developer, production system responsible, manager of IT operations

Maximilia, Avesta, 2009

Business Developer, Board member

Smarkets, London, 2010

Senior Erlang hacker

Ground Rules

Be active

Give comments

Ask questions

What is Erlang? (non-tech version)

- Originally for telephony stuff (for Ericsson) but nowadays used for everything from billing systems to betting exchanges

- Major features are:
  - Easy to do distribution
  - Great error handling

What is Erlang? (tech version)

- Functional language with syntax from Prolog
- Cheap processes (not threads)
- Asynchronous message passing
- Supervisor structure with great handling of errors
- Implementation is Erlang/OTP from Ericsson
- Open source
- Byte compiled (BEAM)
- Machine code compilation available (HiPE)

HiPE

- Created 1996 at Uppsala University
- Distributed with Erlang/OTP since 2001
- Has been stable for many years
- Not officially supported by the OTP team
- A research project
  - 5 register allocators
  - Many modern compiler optimization algorithms
  - Plenty of compiler options
- Supports SPARC, x86, x86_64, powerpc, arm

http://www.it.uu.se/research/group/hipe/

Compiler steps

BEAM
Icode
RTL
ASM
Machine code
Loader

BEAM

Linear code

Icode

From Beam
To Control Flow Graph
Inline Bifs
SSA form optimizations
Dead code elimination
Constant propagation
Some type tests
Lazy code motion
Back to linear code

RTL

From Icode
To CFG
SSA
As Icode
Liveness analysis
More optimizations
To Linear code

ASM

To internal asm language
Machine specific optimizations
Register allocation

Machine code

Runtime system

HiPE compiler (Erlang)
Mode switch Beam/HiPE (asm)
Glue code for bif calls (m4 macro)
Garbage collection (C)
Stubs for BEAM calls (C and asm)
Loader (C and Erlang)
Signal stack handling (C)
Arithmetic overflow (asm and C)

Files

`lib/hipe/`: the HiPE compiler in Erlang
`main/hipe.erl`: the user interface
`main/hipe_main.erl`: main compiler loop
`XXX/hipe_XXX.erl`: data types for XXX

`beam_load.c`: the loader stuff
`erts/emulator/hipe`: C/asm/m4 stuff

Example test file

```
-module(test).
-export([ok/0, test1/1, test2/2]).

ok() ->
  ok.

test1(ok) -> true;
test1(_) -> false.

test2(Value) -> Value =:= ok.
```

Commands

```
>hipe:c({test, ok, 0}, [pp_beam]).
>hipe:c({test, ok, 0}, [pp_icode]).
>hipe:c({test, ok, 0}, [pp_rtl]).
>hipe:c({test, ok, 0}, [pp_asm]).
>hipe:c({test, ok, 0}, [time]).
```

Thank you!

daniel@lunas.se