Polyglot: From the Very Old to the Very New

Chris Seaton
Research Manager
Oracle Labs
July 2017
Safe Harbor Statement

The following is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.
Why people write in Ruby
What about when you reach the limits?
This pure Ruby library can read and write PNG images without depending on an external image library, like RMagick. It tries to be memory efficient and reasonably fast. It supports reading and writing all PNG variants that are defined in the specification, with one limitation: only 8-bit color depth is supported. It supports all transparency, interlacing and filtering options the PNG specifications allows. It can also read and write textual metadata from PNG files. Low-level read/write access to PNG chunks is also possible. This library supports simple drawing on the image canvas and simple operations like alpha composition and cropping. Finally, it can import from and export to RMagick for interoperability. Also, have a look at OilyPNG at http://github.com/wwanbergen/oily_png. OilyPNG is a drop in mixin module that implements some of the ChunkyPNG algorithms in C, which provides a massive speed boost to encoding and decoding.

This Ruby C extension defines a module that can be included into ChunkyPNG to improve its speed.
psd 3.8.0

Parse Photoshop PSD files with ease

psd_native 1.1.3

Native mixins to speed up PSD.rb
def clamp(num, min, max)
    [min, num, max].sort[1]
end

VALUE psd_native_util_clamp(VALUE self,
    VALUE r_num, VALUE r_min, VALUE r_max) {
    int num = FIX2INT(r_num);
    int min = FIX2INT(r_min);
    int max = FIX2INT(r_max);

    return num > max ? r_max : (num < min ? r_min : r_num);
}
Performance on Ruby C Extensions Oily PNG and PSD Native

The technical debt of C extensions
C Extension

C Extension API

Ruby Interpreter
C Extension

C Extension API

JRuby

MRI

Rubinius
Bad news – this isn’t really a thing in practice
C Extension

MRI 3.0

?
String pointers

```c
char *RSTRING_PTR(VALUE string);

static VALUE
ossl_dsa_export(int argc, VALUE *argv, VALUE self)
{
  char *passwd;
  ...
  passwd = RSTRING_PTR(pass);  
  ...
  PEM_write_bio_DSAPrivateKey(out, pkey->pkey.dsa, ciph,  
    NULL, 0, openssl_pem_passwd_cb, passwd)
  ...
}
Array pointers

```c
VALUE *RARRAY_PTR(VALUE array);

VALUE psd_native_blender_compose_bang(VALUE self) {
    ...
    VALUE bg_pixels = rb_funcall(bg_canvas, rb_intern("pixels"), 0);
    VALUE *bg_pixels_ptr = RARRAY_PTR(bg_pixels);
    ...
    for (i = 0, len = RARRAY_LEN(bg_pixels); i < len; i++) {
        bg_pixels_ptr[i] ...
    }
    ...
}
```
Data fields

```c
struct RData {
    struct RBasic basic;
    void (*dmark)(void *data);
    void (*dfree)(void *data);
    void *data;
};

#define RDATA(value) ((struct RData *)value)

#define DATA_PTR(value) (RDATA(value)->data)

static VALUE
ossl_x509req_copy(VALUE self, VALUE other)
{
    ...
    DATA_PTR(self) = X509_REQ_dup(b);
    ...
}
```
Lack of caching when you are in C

```
foo.to_s

method_name = rbIntern("to_s")
rb_funcall(foo, method_name, 0);
```

Last time we called to_s this is the method we used
The black box

def add(a, b)
    a + b
end

add(14, 2)

VALUE add(VALUE self, VALUE a, VALUE b) {
    return INT2FIX(FIX2INT(a) + FIX2INT(b));
}

add(14, 2)
The black box

```
def add(a, b)
    a + b
end

add(14, 2)
```

```
VALUE add(VALUE self, VALUE a, VALUE b) {
    return INT2FIX(FIX2INT(a) + FIX2INT(b));
}

add(14, 2)
```
The black box

```python
def add(a, b)
    a + b
end

add(14, 2) = ?

VALUE add(VALUE self, VALUE a, VALUE b) {
    return INT2FIX(FIX2INT(a) + FIX2INT(b));
}

add(14, 2)
The current workaround to Ruby’s performance problem is now preventing fixing the problem properly
How are people trying to solve this?
Denial

- Everyone should use the FFI or Fiddle
  - FFI and Fiddle are two ways to call C functions directly from Ruby
  - 2.1 billion lines of code in RubyGems, 0.5 billion of it is C extension code
  - It might be nice if people used FFI instead of C extensions... but they don’t... so little point in continuing to argue about it

```ruby
module MyLib
  extend FFI::Library
  ffi_lib 'c'
  attach_function :sqrt, [ :double ], :double
end
```
Bargaining

• Attempt to implement the C extension API as best as possible, alongside optimisations
• Generally involves a lot of copying
• JRuby used this approach in the past, Rubinius still uses it
  – JRuby only ran 60% of C extensions I tried
  – Rubinius ran 90%
  – Worse: when they didn’t work they just ground to a halt, no clear failure point
Bargaining

• Try to improve the C extension API over time
  – The JavaScript (V8) and Java C extension APIs don’t have these problems because they have better designed APIs that don’t expose internals
  – Steady progress in this direction, has helped
  – But even OpenSSL doesn’t use these new methods!

"Don't touch pointers directly"

In MRI (include/ruby/ruby.h), some macros to acquire pointers to the internal data structures are supported such as RARRAY_PTR(), RSTRUCT_PTR() and so on.

DO NOT USE THESE MACROS and instead use the corresponding C-APIs such as rb_ary_arel(), rb_ary_store() and so on."
Depression

• JRuby unfortunately had to give up on their C extension work
  – They didn’t have the resources to maintain it after the original developer moved on
  – Limited compatibility and limited performance
  – In the end, it was removed entirely
  – Maybe it’ll return in the future (they could use the same approach as us)
Acceptance

• JRuby encourage Java extensions instead of C extensions
• Try to optimise Ruby while keeping most of the internals the same
  – IBM’s OMR adds a new GC and JIT to Ruby while keeping support for C extensions
  – The techniques they can use are therefore limited
  – And so performance increases expected from OMR are more modest
Performance on Ruby C Extensions Oily PNG and PSD Native

TruffleRuby
JVM
JVM
Truffle

JVM
The Sulong logo was designed by Valentina Caruso
```c
define i8* @psd_native_util_clamp(i8* %self,
   i8* %r_num, i8* %r_min, i8* %r_max) nounwind uwtable ssp {
   %1 = call i32 @FIX2INT(i8* %r_num)
   %2 = call i32 @FIX2INT(i8* %r_min)
   %3 = call i32 @FIX2INT(i8* %r_max)
   %4 = icmp sgt i32 %1, %3
   br i1 %4, label %5, label %6

   ; <label>:5 ; preds = %0
   br label %12

   ; <label>:6 ; preds = %0
   %7 = icmp slt i32 %1, %2
   br i1 %7, label %8, label %9

   ; <label>:8 ; preds = %6
   br label %10

   ; <label>:9 ; preds = %6
   br label %10

   ; <label>:10 ; preds = %9, %8
   %11 = phi i8* [ %r_min, %8 ], [ %r_num, %9 ]
   br label %12

   ; <label>:12 ; preds = %10, %5
   %13 = phi i8* [ %r_max, %5 ], [ %11, %10 ]
   ret i8* %13
}
```
define i8* @psd_native_util_clamp(i8* %self, 
    i8* %r_num, i8* %r_min, i8* %r_max) nounwind uwtable ssp { 
    %1 = call i32 @FIX2INT(i8* %r_num) 
    %2 = call i32 @FIX2INT(i8* %r_min) 
    %3 = call i32 @FIX2INT(i8* %r_max) 
    %4 = icmp sgt i32 %1, %3 
    br i1 %4, label %5, label %6 
    ; <label>:5 
    br label %12 ; preds = %0 
    ; <label>:6 
    %7 = icmp slt i32 %1, %2 
    br i1 %7, label %8, label %9 
    ; <label>:8 
    br label %10 ; preds = %6 
    ; <label>:9 
    br label %10 ; preds = %6 
    ; <label>:10 
    %11 = phi i8* [%r_min, %8 ], [%r_num, %9 ] 
    br label %12 ; preds = %9, %8 
    ; <label>:12 
    %13 = phi i8* [%r_max, %5 ], [%11, %10 ] 
    ret i8* %13 
}
%4 = icmp sgt i32 %1, %3
br i1 %4, label %5, label %6

; <label>:5
  br label %12

; <label>:6
  %7 = icmp slt i32 %1, %2
  br i1 %7, label %8, label %9
%4 = icmp sgt i32 %1, %3
br i1 %4, label %5, label %6

; <label>:5
    br label %12
; <label>:6
%7 = icmp slt i32 %1, %2
br i1 %7, label %8, label %9

t4 = t1 > t3
if t4
    goto l5
else
    goto l6
end

l5: goto l12
l6: t7 = t1 < t2
if t7
    goto l8
else
    goto l9
end
Ruby and C as two equal languages
Optimise Ruby and C together
Optimise Ruby and C together
Some interesting problems and their solutions
Defining the C extension API in Ruby

```c
int FIX2INT(VALUE value);

int FIX2INT(VALUE value) {
    return truffle_invoke_i(RUBY_CEXT, "FIX2INT", value);
}
```

```ruby
module Truffle::CExt

  def rb_fix2int(value)
    if value.nil?
      raise TypeError
    else
      int = value.to_int
      raise RangeError if int >= 2**32
      int
    end
  end

end
```

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.
Imaginary strings

```c
char *chars = RSTRING_PTR(my_string);
chars[14]
```

Hello, PolyConf!!
Imaginary strings

A Tale of Two String Representations

Kevin Menard - RubyKaigi 2016
Imaginary strings

%1 = call @RSTRING_PTR(%my_string)
%2 = getelementptr %14, 14

char *chars = RSTRING_PTR(my_string);
chars[14]

Hello,

PolyConf!!!
Imaginary strings

```c
char *chars = RSTRING_PTR(my_string);
chars[14]
```

%1 = call @RSTRING_PTR(%my_string)
%2 = getelementptr %14, 14

Hello,

PolyConf!!!
Results
Performance on Ruby C Extensions Oily PNG and PSD Native

Native C extensions give an order of magnitude performance boost

Performance on Ruby C Extensions Oily PNG and PSD Native

Existing attempt to mix managed/native are very disappointing

Performance on Ruby C Extensions Oily PNG and PSD Native

Our solution is 3x faster than native!

It’s clear that cross-language inlining is a key part of the performance.

Limitations
You do need the source code of the C extension

- Means no closed source C extensions
  - Is this a problem in reality for anyone?
  - I’m not aware of any closed source C extensions
  - C extensions in turn using closed source libraries like database drivers is fine
You can’t store pointers to Ruby objects in native code

• If your C extension uses a compiled library, such as libssl.so
  – You can’t give that compiled library a reference to a Ruby object
  – The Ruby object may not really exist
  – The GC may want to move the object

```c
void *rb_jt_to_native_handle(VALUE managed);
VALUE rb_jt_from_native_handle(void *native);

SSL_CTX_set_ex_data(ctx, openssl_ex_ptr_idx, obj);
SSL_CTX_set_ex_data(ctx, openssl_ex_ptr_idx, rb_jt_to_native_handle(obj));
```
To summarise...
Where to find more info
Search for ‘github truffleruby’
Search for ‘github sulong’
Search for ‘oracle graal’
Team

Oracle
Florian Angerer
Danilo Ansaloni
Stefan Anzinger
Martin Balin
Cosmin Basca
Daniele Bonetta
Dušan Bálek
Matthias Brantner
Lucas Braun
Petr Chalupa
Jürgen Christ
Laurent Daynès
Gilles Duboscq
Svatopluk Dědic
Martin Entlicher
Pit Fender
Francois Farquet
Brandon Fish
Matthias Grimmer*
Christian Häubl
Peter Hofer
Bastian Hossbach
Christian Humer
Tomáš Hůrka
Mick Jordan

Oracle (continued)
Vojin Jovanovic
Anantha Kandukuri
Harshad Kasture
Cansu Kaynak
Peter Kessler
Duncan MacGregor
Jiří Maršík
Kevin Menard
Miloslav Metelka
Tomáš Myšák
Petr Pišl
Oleg Plass
Jakub Podlešák
Alessandar Prokopec
Tom Rodriguez
Roland Schatz*
Benjamin Schlegel
Chris Seaton
Jiří Sedláček
Douglas Simon
Štěpán Sindelar
Zbyněk Slajcht
Boris Spasojevic
Lukas Stadler
Codrut Stancu

Oracle (continued)
Jan Štola
Tomáš Stupka
Farhan Tauheed
Jaroslav Tulach
Alexander Ulrich
Michael Van De Vanter
Alessandar Vtorovic
Christian Wimmer
Jürgen Christ
Laurent Daynès
Gilles Duboscq
Svatopluk Dědic
Martin Entlicher
Pit Fender
Francois Farquet
Brandon Fish
Matthias Grimmer*
Christian Häubl
Peter Hofer
Bastian Hossbach
Christian Humer
Tomáš Hůrka
Mick Jordan

Oracle Interns
Brian Belleville
Ondrej Douda
Juan Fumero
Miguel Garcia
Hugo Guiroux
Shams Imam
Berkın Iveyi
Hugo Kapp
Alexey Karyakin
Stephen Kell
Andreas Kunft
Volker Lanting
Gero Leinemann
Julian Lettner
Joe Nash
Tristan Overney
Alessandar Pejovic
David Piorkowski
Philipp Riedmann
Gregor Richards
Robert Seilbeck
Rifat Shariyar

Red Hat
Andrew Dinn
Andrew Haley

Intel
Michael Berg

Twitter
Chris Thalinger

Oracle Alumni
Erik Eckstein
Michael Haupt
Christos Kotselidis
David Leibs
Adam Welc
Till Westmann

JKU Linz
Hanspeter Mössenböck
Bennoi Daloze
Josef Eisl
Thomas Feichtinger
Josef Haider
Christian Huber

Jacob Kreindl*
David Leopoldsdener
Stefan Marr

Thomas Pointhuber*
Manuel Rigger*
Stefan Rumzucker
Bernhard Urban

TU Berlin:
Volker Markl
Andreas Kunft
Jens Meiners
Tilmann Rabl

University of California, Davis
Duncan Temple Lang
Nicholas Uille

T. U. Dortmund
Peter Marwedel
Helena Kotthaus
Ingo Korb

University of California, Irvine
Michael Franz
Yeoul Na
Mohaned Qunaibit
Gulfem Savrun Yeniceri
Wei Zhang

Purdue University
Jan Vitek
Tomas Kalibera
Petr Maj
Lei Zhao

University of Edinburgh
Christophe Dubach
Juan José Fumero Alfonso
Ranjeet Singh
Toomas Remmelg

University of Lugano, Switzerland
Walter Binder
Sun Haiyang

* Team Sulong

Copyright © 2017, Oracle and/or its affiliates. All rights reserved.
Safe Harbor Statement

The preceding is intended to provide some insight into a line of research in Oracle Labs. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. Oracle reserves the right to alter its development plans and practices at any time, and the development, release, and timing of any features or functionality described in connection with any Oracle product or service remains at the sole discretion of Oracle. Any views expressed in this presentation are my own and do not necessarily reflect the views of Oracle.
Integrated Cloud
Applications & Platform Services