Unifying Fixnum and Bignum into Integer

Tanaka Akira
@tanaka_akr
2016-09-08
National Institute of Advanced Industrial Science and Technology (AIST)
Announcement

- Ruby 2.4 unify Fixnum and Bignum into Integer
  [Feature #12005] by naruse

- For Ruby programmers
  - Fixnum class and Bignum class are removed
  - 1.class and (2**100).class returns Integer instead of Fixnum and Bignum
  - The constants, Fixnum and Bignum, references Integer class
    (NameError will not occur)

- For C programmers
  - No internal representation change
  - The global constants, rb_cFixnum and rb_cBignum, are removed
    (compilation error will occur)
  - Class based dispatch doesn't work anymore
    Use FIXNUM_P(obj) and RB_TYPE_P(obj, T_BIGNUM)
  - RUBY_INTEGER_UNIFICATION macro is defined
    (for extension library supporting old and new Ruby)
Fixnum and Bignum

Ruby 2.3 has three classes to represent integers

- **Integer**: abstract super class
  - **Fixnum**: class for small integers
  - **Bignum**: class for big integers

- **Examples**
  - `1.class` => Fixnum
  - `(2**100).class` => Bignum
Integer Unification

Ruby 2.4 has one class to represent integers

- Integer: concrete class
- Examples
  - 1.class => Integer
  - (2**100).class => Integer
Integer Implementations

The range of Fixnum varies

- 32bit CRuby (ILP32): $-2^{30}$ to $2^{30}-1$
- 64bit CRuby (LLP64): $-2^{30}$ to $2^{30}-1$
- 64bit CRuby (LP64): $-2^{62}$ to $2^{62}-1$
- JRuby: $-2^{63}$ to $2^{63}-1$

The range is not portable
Integer Specification

• The specification
  - ISO/IEC 30170:2012 Information technology
    -- Programming languages -- Ruby
  - JIS X 3017:2011 プログラム言語Ruby

• It specifies:
  - There is Integer class which range is unbounded
  - Implementation may (or may not) define subclasses of Integer
  - I.e. Fixnum and Bignum class are not defined (but permitted)

• Ruby 2.3 and Ruby 2.4 conforms the specification

• Conforming program should not depend on Fixnum, Bignum and their range
Fixnum and Bignum is Implementation Detail

- The range of Fixnum varies
  Ruby program should not depend on the range for portability

- The spec. doesn't define Fixnum and Bignum
  Ruby program should not depend on them
Integer Unification for Ruby Programmers

Pros

• Cannot misuse Fixnum and Bignum wrongly
• Learn Ruby easier
• Simplify documents
• Simpler, cleaner, and more mathematical

Cons

• Incompatibility
Wrong Use of Fixnum

lib/rubygems/specification.rb:

```ruby
unless specification_version.is_a?(Fixnum)
  raise Gem::InvalidSpecificationException,
    'specification_version must be a Fixnum (did you mean version?)'
end
```

- This means that valid `specification_version` is:
  - \(-2^{30} \text{ to } 2^{30}-1\) in 32bit CRuby (ILP32),
  - \(-2^{30} \text{ to } 2^{30}-1\) in 64bit CRuby (LLP64),
  - \(-2^{62} \text{ to } 2^{62}-1\) in 64bit CRuby (LP64) and
  - \(-2^{63} \text{ to } 2^{63}-1\) in JRuby.

- This is not the expected meaning
Cannot Use Fixnum and Bignum Wrongly on Ruby 2.4

- Cannot depend the Fixnum range
  `obj.is_a?(Fixnum)` means `obj.is_a?(Integer)`

- REPL doesn't encourage Fixnum
  ```ruby
  irb> 1.class
  => Integer
  ```
  (Fixnum until Ruby 2.3)
Learn Ruby Easier

- Everyone knows 1 is an integer  
  → Not necessary

- Most people doesn't know 1 is a Fixnum (except (Lisp programmers))  
  → Necessary
Simplify Textbooks

- No need to teach Fixnum and Bignum
- Example: Programming Ruby, 2nd edition, p.59

Ruby supports integers and floating-point numbers. Integers can be any length (up to a maximum determined by the amount of free memory on your system). Integers within a certain range (normally $-2^{30}$ to $2^{30}-1$ or $-2^{62}$ to $2^{62}-1$) are held internally in binary form and are objects of class Fixnum. Integers outside this range are stored in objects of class Bignum (currently implemented as a variable-length set of short integers). This process is transparent, and Ruby automatically manages the conversion back and forth.

Ruby supports integers and floating-point numbers. Integers can be any length (up to a maximum determined by the amount of free memory on your system).
Simplify Documents

rdoc forces us to describe document for each method

- Until Ruby 2.3
  - Document Fixnum#foo and Bignum#foo separately
  - It causes duplicated document
  - "ri foo" shows both Fixnum#foo and Bignum#foo
    The duplicated document shown

- Since Ruby 2.4
  - Document only Integer#foo
  - "ri foo" doesn't show duplicated document
Simpleer, Cleaner, and more Mathematical

- Simple
  - Less classes
- Clean
  - No implementation details
- Correspond to mathematics
  - Mathematics: 1 is an integer
  - Ruby: 1.class => Integer
Ruby Level Incompatibility

• Fixnum and Bignum references Integer
• Fixnum range is hidden
• Metaprogramming and DSL
Fixnum and Bignum References Integer

- Fixnum and Bignum references Integer
  
  ```ruby
  irb> Fixnum
  => Integer
  irb> Bignum
  => Integer
  ```

NameError will not occur but

- `Fixnum == Bignum` => true
- `1.is_a?(Bignum)` => true
- `(2**100).is_a?(Fixnum)` => true

- Fixnum and Bignum should not be removed until EOL of Ruby 2.3, at least
Fixnum Range is Hidden

The code for finding Fixnum range will be broken:

- **test/ruby/test_integer_comb.rb:**
  ```ruby
  max = 1
  max *= 2 while (max–1).class == Fixnum
  FIXNUM_MAX = max/2–1
  ```

  This is an infinite loop on Ruby 2.4

- Recommended solution: Don't depend on Fixnum range

- CRuby only non-recommended solution:
  ```ruby
  require 'rbconfig/sizeof' # available since Ruby 2.1
  FIXNUM_MIN = -(1 << (8 * RbConfig::SIZEOF['long'] – 2))
  FIXNUM_MAX = (1 << (8 * RbConfig::SIZEOF['long'] – 2)) – 1
  ```
Metaprogramming and DSL may be Broken

• Metaprogramming
  – mathn.rb defined a method in Fixnum and Bignum
    Solution: Define a method in Integer
  – activesupport prepends NumericWithFormat to Fixnum and Bignum
    Solution: prepends to Integer

• DSL
  – Fixnum and Bignum is not distinguishable
  – Example: Sequel
    http://sequel.jeremyevans.net/rdoc/files/doc/release_notes/4_35_0_txt.html
    DB.create_table(:table) do
      add_column :column, Bignum # Bignum means 64-bit integer column
    end
    Solution: Use a symbol. :Bignum instead of Bignum
Integer Unification for C programmers

Pros

- Nothing

Cons

- Incompatibility
Integer Unification at C-level

- The internal representation is not changed. Fixnum/Bignum distinction still exist at C-level.
- No Fixnum/Bignum class as Ruby-level.
- The global variables for the classes, `rb_cFixnum` and `rb_cBignum`, are not defined. This causes compilation error for problematic extension library.
Integer Representation of Ruby 2.3

Class Hierarchy
- Integer
  - Fixnum
  - Bignum

Fixnum
- pointer 00
- Embedded Bignum
- 64 bits VALUE
  - n 1
- klass
- d0 d1
d2 d3
d4 d5

Bignum
- pointer 00
- Non-embedded Bignum
- 64 bits VALUE
  - len
  - digits

flags 1 embed flag
T_BIG NUM

rb_cInteger
rb_cFixnum
rb_cBignum
Integer Representation of Ruby 2.4

64 bits VALUE

Fixnum

pointer 00

Embedded Bignum

1 flags

klass

d0  d1

d2  d3

d4  d5

Embedded Bignum

Non-embedded Bignum

0 flags

klass

len

digits

Class Hierarchy

rb_cInteger

len
Update Extension Library

• Class comparison:
  - `rb_class_of(obj) == rb_cFixnum` → `FIXNUM_P(obj)`
  - `rb_class_of(obj) == rb_cBignum` → `RB_TYPE_P(obj, T_BIGNUM)`

• Method definition:
  `rb_define_method(rb_cFixnum, "foo", f1, ...)`
  `rb_define_method(rb_cBignum, "foo", f2, ...)`
  →
  ```c
  static VALUE f(...) {
    return FIXNUM_P(self) ? f1(...) : f2(...);
  }
  ```
  `rb_define_method(rb_cInteger, "foo", f, ...)"
Distinguish Ruby 2.3 and 2.4

• RUBY_INTEGER_UNIFICATION macro is defined since Ruby 2.4

• Use following #ifdef
  #ifdef RUBY_INTEGER_UNIFICATION
  code for Ruby 2.4 or later
  #else
  code for Ruby 2.3 or former
  #endif
Affected Extension Libraries

Most libraries to dump/load objects need to be updated

Already modified:

- ext/json
- msgpack [https://github.com/msgpack/msgpack-ruby/pull/115](https://github.com/msgpack/msgpack-ruby/pull/115)
- syck [https://github.com/tenderlove/syck/issues/17](https://github.com/tenderlove/syck/issues/17)
- yajl [https://github.com/chef/ffi-yajl/pull/80](https://github.com/chef/ffi-yajl/pull/80)
- oj [https://github.com/ohler55/oj/issues/305](https://github.com/ohler55/oj/issues/305)
- ox [https://github.com/ohler55/ox/commit/f826190eeeb3cc43f810de5b630a36b78f709dc2](https://github.com/ohler55/ox/commit/f826190eeeb3cc43f810de5b630a36b78f709dc2)
- ruby-gnome2 [https://github.com/ruby-gnome2/ruby-gnome2/commit/dccaf485edd589830e7c37bfde4e78cec147fba](https://github.com/ruby-gnome2/ruby-gnome2/commit/dccaf485edd589830e7c37bfde4e78cec147fba)
- etc.
Version Dependencies

- Several libraries are already modified
- Need to release to use them
- Minor version up (1.3.2 to 1.3.3) is better
- Major version up (1.3.2 to 2.0.0) may cause problems
  Gems with pessimistic version dependency (~> 1.3) must also be updated
- See details for the article by @hsbt
  https://www.hsbt.org/diary/20160829.html#p01
Fixnum/Bignum Agnostic API

- There are few C-level APIs which works Fixnum and Bignum seamlessly
  Example: NUM2LONG, rb_integer_pack/unpack

- More APIs if people wants...
  Why we have no C-level APIs for add/sub/mul/div/mod working both Fixnum and Bignum?
  rb_funcall is good enough for us?
Benefits versus Incompatibility

- The benefits are mostly for beginners
- Fixnum/Bignum is not a big problem for experienced programmers
- Incompatibility annoys people Especially when extension library is not updated immediately
Supported Commiters

- naruse: prevents the wrong use of Fixnum
- akr: simpler documentation
- martin: serious benefits in documentation and learning
- mrkn: more mathematical
- matz: much cleaner & simpler
Summary

- Fixnum and Bignum is unified into Integer at Ruby level
  - Less pitfall
  - Easier learning
  - Simpler documents
- Incompatibility
  - Update your library