Ruby master - Feature #6763

Introduce Flonum technique to speedup floating computation on th 64bit environment

07/21/2012 04:59 AM - ko1 (Koichi Sasada)

Status:	Closed
Priority:	Normal
Assignee:	ko1 (Koichi Sasada)
Target version:	2.0.0

Description

=begin

= Abstract

Introducing Flonum technique, which is similar to Fixnum against Integer, to speedup floating number calculation on the 64bit CPU.

On our measurements, we can achieve x2 performance improvement for simple floating calculation.

= Background

Ruby (CRuby) have a Float class to achieve floating number calculation. However, the Float object is allocated as object every time.

For example,

```
x = 0.0
a = b = 1.2
10.times{
x += (a + b)
}
```

It create over 20 floating number objects.
(a+b) creates one Float object and x+=(a+b) create one more.
This is a one of the biggest reasons why floating number calculation is slow using (C)Ruby.

On the other hands, Fixnum calculation doesn't make new object.

```
x = 0
a = b = 1
10.times{
x += (a + b)
}
```

No object allocated by the above code because Fixnum representation doesn't need object allocation (immediate value).

= Proposal

Introducing Flonum technique for Float such as Fixnum for
Integer on the 64bit CPU enviroment.
A limited ranged Float object can be represented similar way of
Fixnum (immediate object).

Accuracy of the floating calculation is not affected.

We already have tried preliminary implementation and evaluation [1].
In article [1], I describe a technique to introduce Flonum into
CRuby on 64bit CPU environment.

Key idea of our technique is represent a small Float object (mantissa is small enough)
in special bit pattern.
Otherwise, a big float numbers are represented by current object representation.

[1] A Lightweight Representation of Floting-Point Numbers on Ruby Interpreter
http://www.atdot.net/~ko1/activities/rubyfp2008.pdf
http://www.atdot.net/~ko1/activities/rubyfp2008_PPL2008.pdf

Sorry, they are written in Japanese.
Note that we need to change the technique described in this article because proposed technique uses the Fixnum bit pattern for Flonum.

On our implementation, class of Flonum object is Float class.
It is different relation from the relation between Fixnum and Integer.
I think it is a point to discuss.

I heard that MacRuby has similar optimization.

= Compatibility issue

(1) A result of float calculation always return different object if it is same result

a = 1.1 + 1.2
b = 1.1 + 1.2
p(a.object_id == b.object_id) #=> false

After introducing flonum, it will be same objects.

(2) Breaking binary compatibility for C extension

Now, Float object is always `struct RFloat' data type.
After introducing Flonum, the assumption will be break.

=end

Related issues:

Related to Ruby master - Feature #6936: Forbid singleton class and instance v...	**Closed**	**08/27/2012**

History

#1 - 07/21/2012 08:29 AM - Anonymous

Hi,

In message "Re: [ruby-core:46577] [ruby-trunk - Feature #6763][Open] Introduce Flonum technique to speedup floating computation on th 64bit environment"
on Sat, 21 Jul 2012 04:59:05 +0900, "ko1 (Koichi Sasada)" redmine@ruby-lang.org writes:

|= Compatibility issue
|
|(1) A result of float calculation always return different object if it is same result
|
| a = 1.1 + 1.2
| b = 1.1 + 1.2
| p(a.object_id == b.object_id) #=> false
|
|After introducing flonum, it will be same objects.

How do you implement object_id for non integer immediate values?

matz.

#2 - 07/25/2012 06:51 PM - ko1 (Koichi Sasada)

> How do you implement object_id for non integer immediate values?

Good point. It will be considered.

Akr-san proposed that object id of Flonum object should be Bignum (enough big value), because it should be very rare case to use Flonum object id.

#3 - 08/10/2012 03:55 PM - ko1 (Koichi Sasada)

I attached my patch.

Before (32bit CPU):
3.times{
p ObjectSpace._id2ref(p (0.1+0.2).object_id)

}
#=>
20932740
0.30000000000000004
20932680
0.30000000000000004
20932630
0.30000000000000004

After (64bit CPU):
-100880631653099102
0.30000000000000004
-100880631653099102
0.30000000000000004
-100880631653099102
0.30000000000000004

Performance: http://www.atdot.net/fp_store/f.rr1j8m/file.graph.png

clean is current trunk. flonum is modified one. Evaluated on Linux 2.6.32-5-amd64, Intel(R) Xeon(R) CPU E5335 @ 2.00GHz. Higher on Y-axis is good (execution time ratio).

patch: http://www.atdot.net/sp/raw/jw1j8m

#4 - 08/10/2012 04:23 PM - akr (Akira Tanaka)

2012/7/25 ko1 (Koichi Sasada) redmine@ruby-lang.org:

> > How do you implement object_id for non integer immediate values?
>
> Good point. It will be considered.
>
> Akr-san proposed that object id of Flonum object should be Bignum (enough big value), because it should be very rare case to use Flonum object id.

I'm not certain that I proposed anything about object id of Float.
--
Tanaka Akira

#5 - 08/21/2012 03:23 PM - ko1 (Koichi Sasada)

(2012/07/21 4:59), ko1 (Koichi Sasada) wrote:

> Feature #6763: Introduce Flonum technique to speedup floating computation on th 64bit environment
> https://bugs.ruby-lang.org/issues/6763
>
> Author: ko1 (Koichi Sasada)
> Status: Open
> Priority: Normal
> Assignee:
> Category: core
> Target version: 2.0.0
>
> =begin
>
> = Abstract
>
> Introducing Flonum technique, which is similar to Fixnum against Integer, to speedup floating number calculation on the 64bit CPU.
>
> On our measurements, we can achieve x2 performance improvement for simple floating calculation.

Matz also said that commit it and try it.

I'll merge:
https://github.com/ko1/ruby/tree/flonum

--
// SASADA Koichi at atdot dot net

#6 - 10/27/2012 07:27 AM - ko1 (Koichi Sasada)

- Status changed from Open to Closed

- Assignee set to ko1 (Koichi Sasada)

I already merged.
Thanks.