Ruby - Bug #4962

come back gem_prelude!

07/02/2011 02:18 PM - mame (Yusuke Endoh)

Status:	Closed		
Priority:	Normal		
Assignee:	nobu (Nobuyoshi Nakada)		
Target version:	1.9.3		
ruby -v:	-	**Backport:**	

Description

Hello, rubygems developers

Kosaki-san noticed that 1.9.3 is slower than 1.9.2 on many benchmarks.
http://www.atdot.net/sp/view/5qunnl

I investigated and found that the cause is the lack of gem_prelude.rb.

Loading rubygems seems to create many objects and keep the references
to them. See below:

```
$ ruby -ve 'GC.start; p ObjectSpace.count_objects[:TOTAL]'
ruby 1.9.2p180 (2011-02-18 revision 30909) [i686-linux]
9821

$ ./ruby -ve 'GC.start; p ObjectSpace.count_objects[:TOTAL]'
ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
19638

$ ./ruby --disable-gems -ve 'GC.start; p ObjectSpace.count_objects[:TOTAL]'
ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
9821
```

The number of live objects is proportional to the cost of GC mark phase.
You can actually confirm the performance degradation with the following
benchmark script:

```
require 'tempfile'
max = 200_000
str = "Hello world!  " * 1000
f = Tempfile.new('yarv-benchmark')
f.write str
GC::Profiler.enable
max.times{
f.seek 0
f.read
}
p GC::Profiler.total_time
```

```
$ time ruby -v bm_io_file_read.rb
ruby 1.9.2p180 (2011-02-18 revision 30909) [i686-linux]
0.7280460000000308

real    0m3.965s
user    0m2.940s
sys     0m1.024s

$ time ./ruby -v bm_io_file_read.rb
ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
1.396088000000029

real    0m4.786s
user    0m3.716s
```

sys 0m1.060s

$ time ./ruby --disable-gems -v bm_io_file_read.rb
ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
0.7640390000000309

real 0m4.079s
user 0m2.872s
sys 0m1.192s

The performance degradation can be seen by not only such micro benckmarks,
but also my puzzle solvers :-(

There are some approaches to address the problem:

1. to introduce a generational GC; this is impossible until 2.0 because
   it requires modifications to all extension libraries.

2. to diet rubygems; do not create any string, array, hash, and any
   object as much as possible, and do not keep the references to them.

3. to restore gem_prelude.rb to delay loading rubygems.

I guess that 3 is a reasonable choice for 1.9.3. But I'm fine with any
solution to fix rubygems if 1.9.3 becomes as fast as 1.9.2 on the
benchmarks.

--
Yusuke Endoh mame@tsg.ne.jp

History

#1 - 07/02/2011 11:13 PM - luislavena (Luis Lavena)

Yusuke Endoh wrote:

> There are some approaches to address the problem:
>
> 1. to introduce a generational GC; this is impossible until 2.0 because
>   it requires modifications to all extension libraries.
>
> 2. to diet rubygems; do not create any string, array, hash, and any
>   object as much as possible, and do not keep the references to them.
>
> 3. to restore gem_prelude.rb to delay loading rubygems.
>
> I guess that 3 is a reasonable choice for 1.9.3. But I'm fine with any
> solution to fix rubygems if 1.9.3 becomes as fast as 1.9.2 on the
> benchmarks.

AFAIK, The issue with gem_prelude in the past has been that it loaded by default the latest version of every gem in $LOAD_PATH, not allowing you at later time decide another version.

1.9.3 is in feature freeze, right? I was hoping Eric Hodel's proposal desscribed in [ruby-core:31885] could be implemented.

--
Luis Lavena

#2 - 07/02/2011 11:53 PM - kosaki (Motohiro KOSAKI)

- ruby -v changed from ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux] to -

Hi

2011/7/2 Luis Lavena luislavena@gmail.com:

> Issue #4962 has been updated by Luis Lavena.

> Yusuke Endoh wrote:
>
> > There are some approaches to address the problem:
> >
> > Â 1. to introduce a generational GC; this is impossible until 2.0 because
> > Â Â Â it requires modifications to all extension libraries.
> >
> > Â 2. to diet rubygems; do not create any string, array, hash, and any
> > Â Â Â object as much as possible, and do not keep the references to them.
> >
> > Â 3. to restore gem_prelude.rb to delay loading rubygems.
> >
> > I guess that 3 is a reasonable choice for 1.9.3. Â But I'm fine with any
> > solution to fix rubygems if 1.9.3 becomes as fast as 1.9.2 on the
> > benchmarks.
>
> AFAIK, The issue with gem_prelude in the past has been that it loaded by default the latest version of every gem in $LOAD_PATH, not allowing you at later time decide another version.
>
> 1.9.3 is in feature freeze, right? I was hoping Eric Hodel's proposal desscribed in [ruby-core:31885] could be implemented.

Right. but I think this large degression can be considered blocker.
I've compared 192, 192 w/o gem, trunk, trunk w/o gems by following way.

/usr/bin/ruby ../benchmark/driver.rb -v
--executables

#3 - 07/06/2011 06:59 AM - tenderlovemaking (Aaron Patterson)

On Sat, Jul 02, 2011 at 02:18:35PM +0900, Yusuke Endoh wrote:

> Issue #4962 has been reported by Yusuke Endoh.
>
> ---
>
> Bug #4962: come back gem_prelude!
> http://redmine.ruby-lang.org/issues/4962
>
> Author: Yusuke Endoh
> Status: Open
> Priority: Normal
> Assignee: Eric Hodel
> Category: lib
> Target version: 1.9.3
> ruby -v: ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
>
> Hello, rubygems developers
>
> Kosaki-san noticed that 1.9.3 is slower than 1.9.2 on many benchmarks.
> http://www.atdot.net/sp/view/5qunnl
>
> I investigated and found that the cause is the lack of gem_prelude.rb.
>
> Loading rubygems seems to create many objects and keep the references
> to them. See below:
>
> $ ruby -ve 'GC.start; p ObjectSpace.count_objects[:TOTAL]'
> ruby 1.9.2p180 (2011-02-18 revision 30909) [i686-linux]
> 9821
>
> $ ./ruby -ve 'GC.start; p ObjectSpace.count_objects[:TOTAL]'
> ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
> 19638
>
> $ ./ruby --disable-gems -ve 'GC.start; p ObjectSpace.count_objects[:TOTAL]'
> ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
> 9821
>
> The number of live objects is proportional to the cost of GC mark phase.
> You can actually confirm the performance degradation with the following
> benchmark script:
>
> require 'tempfile'
> max = 200_000

```
str = "Hello world!  " * 1000
f = Tempfile.new('yarv-benchmark')
f.write str
GC::Profiler.enable
max.times{
f.seek 0
f.read
}
p GC::Profiler.total_time
```

```
$ time ruby -v bm_io_file_read.rb
ruby 1.9.2p180 (2011-02-18 revision 30909) [i686-linux]
0.7280460000000308

real    0m3.965s
user    0m2.940s
sys     0m1.024s

$ time ./ruby -v bm_io_file_read.rb
ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
1.396088000000029

real    0m4.786s
user    0m3.716s
sys     0m1.060s

$ time ./ruby --disable-gems -v bm_io_file_read.rb
ruby 1.9.3dev (2011-07-01 trunk 32356) [i686-linux]
0.7640390000000309

real    0m4.079s
user    0m2.872s
sys     0m1.192s
```

The performance degradation can be seen by not only such micro benckmarks, but also my puzzle solvers :-(

There are some approaches to address the problem:

1. to introduce a generational GC; this is impossible until 2.0 because it requires modifications to all extension libraries.
2. to diet rubygems; do not create any string, array, hash, and any object as much as possible, and do not keep the references to them.
3. to restore gem_prelude.rb to delay loading rubygems.

We can also help rbconfig go on a diet. I know it's not enough, but this eliminated ~400 object allocations on my machine. (what is the MAKEFILE_CONFIG for anyway?)

```
diff --git a/tool/mkconfig.rb b/tool/mkconfig.rb
index a2221f0..d78a347 100755
--- a/tool/mkconfig.rb
+++ b/tool/mkconfig.rb
@@ -202,8 +202,6 @@ print <<EOS
CONFIG["vendorlibdir"] = "$(vendordir)/$(ruby_version)"
CONFIG["vendorarchdir"] = "$(vendorlibdir)/$(sitearch)"
CONFIG["topdir"] = File.dirname(__FILE__)
```

- MAKEFILE_CONFIG = {}
- CONFIG.each{|k,v| MAKEFILE_CONFIG[k] = v.dup}
  def RbConfig::expand(val, config = CONFIG)
  newval = val.gsub(/\$\$|\$\(([^()]+)\)|\$\{([^{}]+)\}/) {
  var = $&
  @@ -233,6 +231,13 @@ print <<EOS
  RbConfig::CONFIG["ruby_install_name"] + RbConfig::CONFIG["EXEEXT"]
)
  end

- def self.const_missing const
- return super unless :MAKEFILE_CONFIG == const
- const_set :MAKEFILE_CONFIG, {}

- CONFIG.each{|k,v| MAKEFILE_CONFIG[k] = v.dup}
- MAKEFILE_CONFIG
- end
  end
  autoload :Config, "rbconfig/obsolete.rb" # compatibility for ruby-1.8.4 and older.
  CROSS_COMPILING = nil unless defined? CROSS_COMPILING

--
Aaron Patterson
http://tenderlovemaking.com/

#4 - 07/06/2011 07:23 AM - luislavena (Luis Lavena)

On Tue, Jul 5, 2011 at 6:56 PM, Aaron Patterson
aaron@tenderlovemaking.com wrote:

> We can also help rbconfig go on a diet. Â I know it's not enough, but
> this eliminated ~400 object allocations on my machine. Â (what is the
> MAKEFILE_CONFIG for anyway?)

Is used by mkmf, which raises another round of questions: why is not using RbConfig::CONFIG directly?

Luis Lavena
AREA 17

Perfection in design is achieved not when there is nothing more to add,
but rather when there is nothing more to take away.
Antoine de Saint-ExupÃ©ry

#5 - 07/06/2011 07:23 AM - tenderlovemaking (Aaron Patterson)

On Wed, Jul 06, 2011 at 07:01:37AM +0900, Luis Lavena wrote:

> On Tue, Jul 5, 2011 at 6:56 PM, Aaron Patterson
> aaron@tenderlovemaking.com wrote:
>
> > We can also help rbconfig go on a diet. I know it's not enough, but
> > this eliminated ~400 object allocations on my machine. (what is the
> > MAKEFILE_CONFIG for anyway?)
>
> Is used by mkmf, which raises another round of questions: why is not
> using RbConfig::CONFIG directly?

I assume it's so that if someone mutates the hash, it doesn't impact the
RbConfig::CONFIG hash. Though, if that's the case, why doesn't mkmf.rb
just dup the hash rather than relying on rbconfig.

Maybe this patch is more appropriate:

diff --git a/lib/mkmf.rb b/lib/mkmf.rb
index 9b0b8c7..efaa603 100644
--- a/lib/mkmf.rb
+++ b/lib/mkmf.rb
@@ -6,7 +6,7 @@ require 'rbconfig'
require 'fileutils'
require 'shellwords'

-CONFIG = RbConfig::MAKEFILE_CONFIG
+CONFIG = RbConfig.makefile_config
ORIG_LIBPATH = ENV['LIB']

C_EXT = %w[c m]
diff --git a/template/fake.rb.in b/template/fake.rb.in
index 7bfa0ae..b487a79 100755
--- a/template/fake.rb.in
+++ b/template/fake.rb.in
@@ -24,7 +24,7 @@ end

$:.unshift(File.expand_path("..", **FILE**))

posthook = proc do

- mkconfig = RbConfig::MAKEFILE_CONFIG

- mkconfig = RbConfig.makefile_config
  extout = File.expand_path(mkconfig["EXTOUT"], mkconfig["builddir"])
  $arch_hdrdir = "#{extout}/include/$(arch)"
  $ruby = baseruby
  @@ -33,7 +33,7 @@ end
  prehook = proc do |extmk|
  unless extmk
  config = RbConfig::CONFIG

- mkconfig = RbConfig::MAKEFILE_CONFIG

- mkconfig = RbConfig.makefile_config
  builddir = File.expand_path(File.dirname(**FILE**))
  mkconfig["top_srcdir"] = $top_srcdir = File.expand_path("@top_srcdir@", builddir)
  mkconfig["rubyhdrdir"] = "$(top_srcdir)/include"
  diff --git a/tool/compile_prelude.rb b/tool/compile_prelude.rb
  index 6ad9fce..0674754 100755
  --- a/tool/compile_prelude.rb
  +++ b/tool/compile_prelude.rb
  @@ -49,9 +49,9 @@ class Prelude
  key = $1
  unless @mkconf
  require './rbconfig'

- [redacted]

- [redacted]
  ```
  end
  ```

- [redacted]

- [redacted]
  ```
  val = RbConfig.expand("$(#{key})", @mkconf)
  @need_ruby_prefix ||= /\A\#\{TMP_RUBY_PREFIX\}/ =~ val
  c_esc(val)
  ```

diff --git a/tool/mkconfig.rb b/tool/mkconfig.rb
index a2221f0..e924696 100755
--- a/tool/mkconfig.rb
+++ b/tool/mkconfig.rb
@@ -202,8 +202,6 @@ print <<EOS
CONFIG["vendorlibdir"] = "$(vendordir)/$(ruby_version)"
CONFIG["vendorarchdir"] = "$(vendorlibdir)/$(sitearch)"
CONFIG["topdir"] = File.dirname(**FILE**)

- MAKEFILE_CONFIG = {}
- CONFIG.each{|k,v| MAKEFILE_CONFIG[k] = v.dup}
  def RbConfig::expand(val, config = CONFIG)
  newval = val.gsub(/\$\$|\$\(([^()]+)\)|\$\{([^{}]+)\}/) {
  var = $&
  @@ -233,6 +231,17 @@ print <<EOS
  RbConfig::CONFIG["ruby_install_name"] + RbConfig::CONFIG["EXEEXT"]
)
  end

- def self.makefile_config
- hash = CONFIG.dup
- hash.each { |k,v| hash[k] = v.dup }
- end

- def self.const_missing const
- return super unless :MAKEFILE_CONFIG == const
- const_set :MAKEFILE_CONFIG, makefile_config
- MAKEFILE_CONFIG
- end
  end
  autoload :Config, "rbconfig/obsolete.rb" # compatibility for ruby-1.8.4 and older.

CROSS_COMPILING = nil unless defined? CROSS_COMPILING
diff --git a/win32/Makefile.sub b/win32/Makefile.sub
index 0b1361a..34be069 100644
--- a/win32/Makefile.sub
+++ b/win32/Makefile.sub
@@ -968,7 +968,7 @@ $(ruby_pc): $(RBCONFIG)
@$(MINIRUBY) -r./rbconfig -p
-e 'STDOUT.binmode'
-e '$$.gsub!(/@([a-z]\w*)@/i) {' \

- -e '(RbConfig::MAKEFILE_CONFIG[$$1] or "").gsub(/$$((.+?))/, %Q[$${\1}])' \
- -e '(RbConfig::CONFIG[$$1] or "").gsub(/$$((.+?))/, %Q[$${\1}])'
-e '}'
$(srcdir)/template/ruby.pc.in > $@

diff --git a/win32/resource.rb b/win32/resource.rb
index 786edb0..34a1df2 100755
--- a/win32/resource.rb
+++ b/win32/resource.rb
@@ -2,7 +2,7 @@

require './rbconfig'

-CONFIG = RbConfig::MAKEFILE_CONFIG
+CONFIG = RbConfig.makefile_config

version = RUBY_VERSION.split(/./)
patch = CONFIG['PATCHLEVEL']

--
Aaron Patterson
http://tenderlovemaking.com/

#6 - 07/06/2011 08:46 AM - drbrain (Eric Hodel)

I have found some unnecessary work in rubygems and am running benchmarks to see what effect delaying them until they're necessary have on make benchmark.

#7 - 07/06/2011 09:23 AM - kosaki (Motohiro KOSAKI)

> I have found some unnecessary work in rubygems and am running benchmarks to see what effect delaying them until they're necessary have on make benchmark.

Great!
Do we have any chance to get your improvement until 1.9.3 release?

#8 - 07/06/2011 11:06 AM - drbrain (Eric Hodel)

I need to verify both correctness of the behavior of RubyGems and speed improvements in make benchmark.

I should be able to commit my changes and report my findings tomorrow.

It would help if someone could comment on Aaron's rbconfig.rb patch, I do not have enough experience to commit it, but it should help increase startup speed.

#9 - 07/06/2011 01:13 PM - drbrain (Eric Hodel)

- Status changed from Open to Assigned

I have made three runs of make benchmark using the following revisions of ruby:

ruby 1.9.2p180 (2011-02-18 revision 30909) [x86_64-darwin10.8.0]

ruby 1.9.3dev (2011-07-05 trunk 32413) [x86_64-darwin10.8.0]

The benchmark bm_vm_thread_mutex3.rb was disabled as it presented an an extreme outlier for 1.9.2p180.

I took the total time it took all benchmarks to run.

The first run is with the ruby checkout

1.9.2: 204.890 206.312 209.319
1.9.3: 210.793 215.815 214.773
diff: 5.903 9.503 5.454

(For diff, smaller is better)

The second run is with --disable-gems for 1.9.3. I modified RUNRUBY in Makefile:

RUNRUBY = $(MINIRUBY) $(srcdir)/tool/runruby.rb --extout=$(EXTOUT) $(RUNRUBYOPT) -- --disable-gems

1.9.2: 215.472 206.452 205.110
1.9.3: 201.837 194.694 191.747
diff: -13.635 -11.758 -13.363

The third run is with my changes to delay work in rubygems.rb:

1.9.2: 208.982 211.249 208.637
1.9.3: 198.714 201.984 198.293
diff: -10.268 -9.265 -10.344

Here are the average differences from 1.9.2-p180:

stock ruby trunk: 6.953
--disable-gems: -12.919
rubygems patches: -9.959

Is the slowdown of 2.96 seconds between --disable-gems and my fixes across all benchmarks acceptable?

Should I look for additional improvements?

#10 - 07/06/2011 06:53 PM - kosaki (Motohiro KOSAKI)

Hi

Nice improvement!

> Issue #4962 has been updated by Eric Hodel.
>
> Status changed from Open to Assigned
>
> I have made three runs of make benchmark using the following revisions of ruby:
>
> ruby 1.9.2p180 (2011-02-18 revision 30909) [x86_64-darwin10.8.0]
>
> ruby 1.9.3dev (2011-07-05 trunk 32413) [x86_64-darwin10.8.0]
>
> The benchmark bm_vm_thread_mutex3.rb was disabled as it presented an an extreme outlier for 1.9.2p180.

Ah, yes. This is the reason why I rewrote GVL. We should ignore it.

> I took the total time it took all benchmarks to run.
>
> The first run is with the ruby checkout
>
> 1.9.2: Â 204.890 206.312 209.319
> 1.9.3: Â 210.793 215.815 214.773
> diff: Â 5.903 Â 9.503 Â 5.454
>
> (For diff, smaller is better)
>
> The second run is with --disable-gems for 1.9.3. Â I modified RUNRUBY in Makefile:
>
> RUNRUBY

#11 - 07/06/2011 08:59 PM - mame (Yusuke Endoh)

Hello,

2011/7/6 KOSAKI Motohiro kosaki.motohiro@gmail.com:

> Anyway, personally I think it is acceptable and no more improvemnt
> because usually people only

> compare 1.9.2 and 1.9.3 and don't compare individual patches in 1.9.3 changes.
>
> Endoh-san, What do you think?

Looks very good. I'd like to try it myself. Eric, could you
show the patch? Or, you may commit it directly to ruby trunk
if you think the patch is simple.

--
Yusuke Endoh mame@tsg.ne.jp

#12 - 07/07/2011 03:23 AM - drbrain (Eric Hodel)

On Jul 6, 2011, at 2:48 AM, KOSAKI Motohiro wrote:

> > Here are the average differences from 1.9.2-p180:
> >
> > stock ruby trunk: 6.953
> > --disable-gems: -12.919
> > rubygems patches: -9.959
> >
> > Is the slowdown of 2.96 seconds between --disable-gems and my fixes across all benchmarks acceptable?
> >
> > Should I look for additional improvements?
>
> Great!
>
> Can you please tell us a result of vm3_gc and io_file_read? They have
> most big degressions and I'm worry about it.

io_file_read

--disable-gems: 3.978 3.768 4.007
rubygems patch: 3.944 3.960 4.072

vm3_gc

--disable-gems: 1.166 1.157 1.154
rubygems patch: 1.616 1.630 1.632

I poked at setting RUBY_HEAP_MIN_SLOTS@000 when starting up ruby with rubygems enabled. This allowed ruby to start up without running the garbage collector and didn't affect the resident size of the process much.

I didn't run a full make benchmark to see if it made any larger difference.

> Anyway, personally I think it is acceptable and no more improvemnt
> because usually people only compare 1.9.2 and 1.9.3 and don't compare individual patches in 1.9.3 changes.

#13 - 07/07/2011 03:23 AM - drbrain (Eric Hodel)

On Jul 6, 2011, at 4:57 AM, Yusuke ENDOH wrote:

> 2011/7/6 KOSAKI Motohiro kosaki.motohiro@gmail.com:
>
> > Anyway, personally I think it is acceptable and no more improvemnt
> > because usually people only
> > compare 1.9.2 and 1.9.3 and don't compare individual patches in 1.9.3 changes.
> >
> > Endoh-san, What do you think?
>
> Looks very good. I'd like to try it myself. Eric, could you
> show the patch? Or, you may commit it directly to ruby trunk
> if you think the patch is simple.

The patch is simple, I will commit it after I eat lunch (about 2 hours).

#14 - 07/07/2011 06:21 AM - drbrain (Eric Hodel)

Please try r32429

#15 - 07/07/2011 07:23 AM - Eregon (Benoit Daloze)

On 6 July 2011 23:21, Eric Hodel drbrain@segment7.net wrote:

> Please try r32429

I just tried, and here are some numbers.
Surprising how such a change can move numbers.

$ time ruby -e ''
before: 0.045
no-gems: 0.013
after: 0.024

$ ruby -e 'p ObjectSpace.count_objects[:TOTAL]'
before: 20033
no-gems: 9811
after: 14308

#16 - 07/07/2011 07:29 AM - drbrain (Eric Hodel)

On Jul 6, 2011, at 3:02 PM, Benoit Daloze wrote:

> On 6 July 2011 23:21, Eric Hodel drbrain@segment7.net wrote:
>
> > Please try r32429
>
> I just tried, and here are some numbers.
> Surprising how such a change can move numbers.
>
> $ time ruby -e ''
> before: 0.045
> no-gems: 0.013
> after: 0.024
>
> $ ruby -e 'p ObjectSpace.count_objects[:TOTAL]'
> before: 20033
> no-gems: 9811
> after: 14308

I'm unsure if TOTAL is counting what we think it's counting. It seems to be showing the size of the object space not the number of allocated objects.
If you add allocated and FREE from the output below you'll get TOTAL.

$ cat diffhash.rb
gems

#17 - 07/09/2011 09:36 AM - drbrain (Eric Hodel)

May I close this ticket?

#18 - 07/09/2011 01:20 PM - kosaki (Motohiro KOSAKI)

- Assignee changed from drbrain (Eric Hodel) to nobu (Nobuyoshi Nakada)

Now, nobu is reviewing Aaron's patch. Please don't close it awhile. Instead I reassign the ticket to nobu.

Thanks.

#19 - 07/09/2011 01:34 PM - kosaki (Motohiro KOSAKI)

And, here is laster benchmark comparision on Linux Fedora15 x86-64.

You have to ignore vm_thead_xx and vm3_clearmethodcache. They are influenced another changes.
And only followint three have significant difference.

name	192r31932-nogems	192r31932	trunk-nogems	trunk
app_mandelbrot	3.240	3.274	3.398	4.125
io_file_read	6.756	7.214	7.685	8.123
vm3_gc	2.084	2.167	2.259	3.296

Do anyone have more improvement idea?

Cheeers.

```
%  /usr/bin/ruby ../benchmark/driver.rb -v --executables="192r31932-nogems::~/ruby/bin/ruby-192-r31932 --disable-gems;
192r31932::~/ruby/bin/ruby-192-r31932; trunk-nogems::~/ruby/bin/ruby-trunk --disable-gems; trunk::~/ruby/bin/ruby-trunk -I../lib -I. -I.ext/common
../tool/runruby.rb --extout=.ext  --"  --pattern='bm_' --directory=../benchmark -r 5  Elapesed time: 39930.060974 (sec)
---------------------------------------------------------- benchmark results: minimum results in each 5 measurements. name     192r31932-nogems
192r31932       trunk-nogems  trunk app_answer      0.157   0.154   0.156   0.195    app_erb 2.475   2.484   2.517   2.652 app_factorial   2.558   2.591
2.482   2.737 app_fib 1.833   1.845   1.777   1.906 app_mandelbrot  3.240   3.274   3.398   4.125 app_pentomino   32.655  32.803  33.764  34.966
app_raise       1.146   1.148   1.081   1.189 app_strconcat   2.734   2.808   2.821   2.832 app_tak 2.791   2.787   2.714   2.770 app_tarai       2.153
2.145   2.170   2.225 app_uri 1.633   1.628   1.654   1.767 io_file_create  2.514   2.482   2.035   2.214 io_file_read    6.756   7.214   7.685   8.123
io_file_write   2.206   2.152   2.161   2.254 io_select       2.253   2.283   2.550   2.619 io_select2      6.379   6.478   7.192   6.084 io_select3      0.871
0.847   0.887   0.937 loop_for        3.003   3.036   2.863   2.905 loop_generator  1.031   1.073   0.889   0.945 loop_times      2.606   2.593   2.636
2.753 loop_whileloop  1.608   1.606   1.534   1.265 loop_whileloop2 0.347   0.334   0.335   0.321 so_ackermann    2.034   2.008   2.023   2.062
so_array        2.369   2.375   2.884   2.935 so_binary_trees 0.889   0.913   0.932   0.995 so_concatenate  7.358   7.322   7.271   7.310
so_count_words  0.534   0.535   0.515   0.585 so_exception    2.150   2.172   2.022   2.245 so_fannkuch     2.821   2.809   2.793   3.147 so_fasta
4.160   4.225   4.446   4.784 so_k_nucleotide 3.341   3.400   3.232   3.313 so_lists        1.651   1.675   1.661   1.717 so_mandelbrot   11.325  11.690
11.764  11.530 so_matrix       1.530   1.526   1.542   1.592 so_meteor_contest       9.493   8.700   8.517   9.820 so_nbody        8.184   8.339   8.191
7.797 so_nested_loop  2.527   2.447   2.443   2.491 so_nsieve       7.351   7.252   8.503   8.516 so_nsieve_bits  4.920   4.945   4.896   4.973
so_object       1.779   1.781   1.752   1.750 so_partial_sums 10.433  10.625  10.520  10.269 so_pidigits    1.710   1.750   1.811   1.986 so_random
1.834   1.899   1.958   1.863 so_reverse_complement   3.198   3.351   3.542   3.615 so_sieve        2.141   2.161   2.439   2.489 so_spectralnorm
7.887   8.088   7.805   7.412 vm1_block*      3.863   3.462   3.850   3.953 vm1_const*      0.821   0.860   1.107   0.668 vm1_ensure*     0.951   1.012
1.197   0.973 vm1_ivar*       1.238   1.232   0.879   0.878 vm1_ivar_set*   1.824   1.286   1.126   1.201 vm1_length*     1.389   1.329   1.403   1.260
vm1_neq*        0.921   0.902   1.114   0.831 vm1_not*        0.297   0.315   0.407   0.514 vm1_rescue*     0.140   0.124   0.361   0.134
vm1_simplereturn*       2.389   2.409   2.992   2.523 vm1_swap*       2.000   2.011   2.081   1.033 vm2_array*      1.421   1.463   1.443   1.910
vm2_case*       0.314   0.331   0.322   0.297 vm2_defined_method*     6.192   6.221   6.884   6.565 vm2_eval*       29.320  29.618  30.998  38.323
vm2_method*     3.458   3.479   3.610   3.790 vm2_mutex*      1.748   1.776   2.010   1.905 vm2_poly_method*        5.353   5.345   5.992   5.731
vm2_poly_method_ov*     0.546   0.554   0.530   0.472 vm2_proc*       0.913   0.930   0.984   0.994 vm2_regexp*     2.351   2.379   2.412   2.398
vm2_send*       0.554   0.595   0.500   0.552 vm2_super*      1.119   1.145   1.328   1.244 vm2_unif1*      0.566   0.587   0.599   0.580 vm2_zsuper*
1.242   1.290   1.476   1.385 vm3_clearmethodcache    4.762   4.992   0.822   1.008 vm3_gc  2.084   2.167   2.259   3.296 vm_thread_alive_check1
0.346   0.356   0.436   0.516 vm_thread_create_join   5.791   5.737   5.932   5.915 vm_thread_mutex1        1.568   1.587   1.642   1.700
vm_thread_mutex2        1.596   1.616   5.558   2.638 vm_thread_mutex3        3131.381        1335.436        4.009   4.089 vm_thread_pass  0.130
0.137   1.687   1.862 vm_thread_pass_flood    0.356   0.311   0.705   0.847 vm_thread_pipe  1.154   1.133   2.515   2.459
```

#20 - 07/11/2011 08:53 AM - nobu (Nobuyoshi Nakada)

Hi,

At Wed, 6 Jul 2011 06:56:09 +0900,
Aaron Patterson wrote in [ruby-core:37813]:

> We can also help rbconfig go on a diet. I know it's not enough, but
> this eliminated ~400 object allocations on my machine. (what is the
> MAKEFILE_CONFIG for anyway?)

MAKEFILE_CONFIG keeps make macros unexpanded. So you should
keep MAKEFILE_CONFIG not CONFIG, and expand it dynamically when
the latter is accessed.

--
Nobu Nakada

#21 - 07/11/2011 08:56 AM - kosaki (Motohiro KOSAKI)

- Status changed from Assigned to Closed

> MAKEFILE_CONFIG keeps make macros unexpanded. So you should
> keep MAKEFILE_CONFIG not CONFIG, and expand it dynamically when
> the latter is accessed.

Then, I'll close this ticket.
Aaron, if you have a motivation of a respin, could you please create a new ticket ?
This thread is already too long and have multiple topics.

#22 - 07/13/2011 02:53 PM - tenderlovemaking (Aaron Patterson)

On Mon, Jul 11, 2011 at 08:50:12AM +0900, Nobuyoshi Nakada wrote:

> Hi,

> At Wed, 6 Jul 2011 06:56:09 +0900,
> Aaron Patterson wrote in [ruby-core:37813]:
>
> > We can also help rbconfig go on a diet. I know it's not enough, but
> > this eliminated ~400 object allocations on my machine. (what is the
> > MAKEFILE_CONFIG for anyway?)
>
> MAKEFILE_CONFIG keeps make macros unexpanded. So you should
> keep MAKEFILE_CONFIG not CONFIG, and expand it dynamically when
> the latter is accessed.

I see. What did you think of my patch that only defined MAKEFILE_CONFIG during const_missing? That /should/ result in the same behavior (I think).

--
Aaron Patterson
http://tenderlovemaking.com/

Files

noname	500 Bytes	07/06/2011	tenderlovemaking (Aaron Patterson)
noname	500 Bytes	07/06/2011	tenderlovemaking (Aaron Patterson)
noname	500 Bytes	07/13/2011	tenderlovemaking (Aaron Patterson)