Ruby - Feature #19236

Allow to create hashes with a specific capacity from Ruby

12/15/2022 09:02 AM - byroot (Jean Boussier)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

Followup on [Feature #18683] which added a C-API for this purpose.

Various protocol parsers such as Redis RESP3 or msgpack, have to create hashes, and they know the size in advance.
For efficiency, it would be preferable if they could directly allocate a Hash of the necessary size, so that large hashes wouldn't cause many re-alloccations and re-hash.

String and Array both already offer similar APIs:

```
String.new(capacity: XXX)
Array.new(XX) / rb_ary_new_capa(long)
```

However there's no such public API for Hashes in Ruby land.

Proposal

I think Hash should have a way to create a new hash with a capacity parameter.

The logical signature of Hash.new(capacity: 1000) was deemed too incompatible in [Feature #18683].

@Eregon (Benoit Daloze) proposed to add Hash.create(capacity: 1000).

Related issues:

Related to Ruby - Feature #18683: Allow to create hashes with a specific capa...	**Closed**

Associated revisions

Revision 31ac8efca8ecb574e1e7b7c32cce54cb1b97f19a - 05/23/2023 01:51 PM - Jean byroot Boussier

Hash.new: print a deprecation warning when receiving keyword arguments (#7828)

[Feature #19236]

In Ruby 3.3, Hash.new shall print a deprecation warning if keyword arguments
are passed instead of treating them as an implicit positional Hash.

This will allow to safely introduce a capacity keyword argument in 3.4

Co-authored-by: Jean Boussier byroot@ruby-lang.org

Revision 31ac8efca8ecb574e1e7b7c32cce54cb1b97f19a - 05/23/2023 01:51 PM - Jean byroot Boussier

Hash.new: print a deprecation warning when receiving keyword arguments (#7828)

[Feature #19236]

In Ruby 3.3, Hash.new shall print a deprecation warning if keyword arguments
are passed instead of treating them as an implicit positional Hash.

This will allow to safely introduce a capacity keyword argument in 3.4

Co-authored-by: Jean Boussier byroot@ruby-lang.org

Revision 31ac8efc - 05/23/2023 01:51 PM - Jean byroot Boussier

Hash.new: print a deprecation warning when receiving keyword arguments (#7828)

[Feature #19236]

In Ruby 3.3, Hash.new shall print a deprecation warning if keyword arguments are passed instead of treating them as an implicit positional Hash.

This will allow to safely introduce a capacity keyword argument in 3.4

Co-authored-by: Jean Boussier byroot@ruby-lang.org

Revision 9594db0cf28d7bc10bfc46142239191a11f1dbbe - 07/08/2024 10:24 AM - byroot (Jean Boussier)

Implement Hash.new(capacity:)

[Feature #19236]

When building a large hash, pre-allocating it with enough capacity can save many re-hashes and significantly improve performance.

```
/opt/rubies/3.3.0/bin/ruby --disable=gems -rrubygems -I./benchmark/lib ./benchmark/benchmark-driver/exe/benchm
ark-driver \
                    --executables="compare-ruby::../miniruby-master -I.ext/common --disable-gem" \
                    --executables="built-ruby::./miniruby --disable-gem" \
                    --output=markdown --output-compare -v $(find ./benchmark -maxdepth 1 -name 'hash_new' -o -
name '*hash_new*.yml' -o -name '*hash_new*.rb' | sort)
compare-ruby: ruby 3.4.0dev (2024-03-25T11:48:11Z master f53209f023) +YJIT dev [arm64-darwin23]
last_commit=[ruby/irb] Cache RDoc::RI::Driver.new (https://github.com/ruby/irb/pull/911)
built-ruby: ruby 3.4.0dev (2024-03-25T15:29:40Z hash-new-rb 77652b08a2) +YJIT dev [arm64-darwin23]
warming up...

|                    |compare-ruby|built-ruby|
|:-------------------|-----------:|---------:|
|new                 |      7.614M|    5.976M|
|                    |       1.27x|         -|
|new_with_capa_1k    |     13.931k|   15.698k|
|                    |           -|     1.13x|
|new_with_capa_100k  |     124.746|   148.283|
|                    |           -|     1.19x|
```

Revision 9594db0cf28d7bc10bfc46142239191a11f1dbbe - 07/08/2024 10:24 AM - byroot (Jean Boussier)

Implement Hash.new(capacity:)

[Feature #19236]

When building a large hash, pre-allocating it with enough capacity can save many re-hashes and significantly improve performance.

```
/opt/rubies/3.3.0/bin/ruby --disable=gems -rrubygems -I./benchmark/lib ./benchmark/benchmark-driver/exe/benchm
ark-driver \
                    --executables="compare-ruby::../miniruby-master -I.ext/common --disable-gem" \
                    --executables="built-ruby::./miniruby --disable-gem" \
                    --output=markdown --output-compare -v $(find ./benchmark -maxdepth 1 -name 'hash_new' -o -
name '*hash_new*.yml' -o -name '*hash_new*.rb' | sort)
compare-ruby: ruby 3.4.0dev (2024-03-25T11:48:11Z master f53209f023) +YJIT dev [arm64-darwin23]
last_commit=[ruby/irb] Cache RDoc::RI::Driver.new (https://github.com/ruby/irb/pull/911)
built-ruby: ruby 3.4.0dev (2024-03-25T15:29:40Z hash-new-rb 77652b08a2) +YJIT dev [arm64-darwin23]
warming up...

|                    |compare-ruby|built-ruby|
|:-------------------|-----------:|---------:|
|new                 |      7.614M|    5.976M|
|                    |       1.27x|         -|
|new_with_capa_1k    |     13.931k|   15.698k|
|                    |           -|     1.13x|
|new_with_capa_100k  |     124.746|   148.283|
|                    |           -|     1.19x|
```

Revision 9594db0c - 07/08/2024 10:24 AM - byroot (Jean Boussier)

Implement Hash.new(capacity:)

[Feature #19236]

When building a large hash, pre-allocating it with enough capacity can save many re-hashes and significantly improve

performance.

```
/opt/rubies/3.3.0/bin/ruby --disable=gems -rrubygems -I./benchmark/lib ./benchmark/benchmark-driver/exe/benchmark-driver \
                    --executables="compare-ruby::../miniruby-master -I.ext/common --disable-gem" \
                    --executables="built-ruby::./miniruby --disable-gem" \
                    --output=markdown --output-compare -v $(find ./benchmark -maxdepth 1 -name 'hash_new' -o -name '*hash_new*.yml' -o -name '*hash_new*.rb' | sort)
compare-ruby: ruby 3.4.0dev (2024-03-25T11:48:11Z master f53209f023) +YJIT dev [arm64-darwin23]
last_commit=[ruby/irb] Cache RDoc::RI::Driver.new (https://github.com/ruby/irb/pull/911)
built-ruby: ruby 3.4.0dev (2024-03-25T15:29:40Z hash-new-rb 77652b08a2) +YJIT dev [arm64-darwin23]
warming up...

|                     |compare-ruby|built-ruby|
|:--------------------|-----------:|---------:|
|new                  |      7.614M|    5.976M|
|                     |       1.27x|         -|
|new_with_capa_1k     |     13.931k|   15.698k|
|                     |           -|     1.13x|
|new_with_capa_100k   |     124.746|   148.283|
|                     |           -|     1.19x|
```

Revision 796d315958f9d23ca3b3524f76c4f267c65edfe0 - 11/26/2024 10:46 AM - byroot (Jean Boussier)

Add NEWS.md entry about Hash.new(capacity:)

[Feature #19236]

Ref: https://github.com/ruby/ruby/pull/10357

Revision 796d315958f9d23ca3b3524f76c4f267c65edfe0 - 11/26/2024 10:46 AM - byroot (Jean Boussier)

Add NEWS.md entry about Hash.new(capacity:)

[Feature #19236]

Ref: https://github.com/ruby/ruby/pull/10357

Revision 796d3159 - 11/26/2024 10:46 AM - byroot (Jean Boussier)

Add NEWS.md entry about Hash.new(capacity:)

[Feature #19236]

Ref: https://github.com/ruby/ruby/pull/10357

History

#1 - 12/15/2022 09:02 AM - byroot (Jean Boussier)

- Related to Feature #18683: Allow to create hashes with a specific capacity. added

#2 - 12/25/2022 05:16 PM - janosch-x (Janosch Müller)

maybe the genie is out of the bottle already, but it would be nice to have a uniform API for creating objects with a given capacity, e.g.

```
Array.with_capacity(100)       # => []
Hash.with_capacity(100)        # => {}
IO::Buffer.with_capacity(100) # => #<IO::Buffer>
String.with_capacity(100)      # => ''
# more?
```

for Array and IO::Buffer, ::with_capacity would essentially be an alias for ::new. for String, the capacity kwarg could be deprecated to limit the number of APIs.

#3 - 01/20/2023 05:31 AM - mame (Yusuke Endoh)

Discussed at the dev meeting.

@matz (Yukihiro Matsumoto) said that Hash.create(capacity: 4096) is acceptable (unless it conflicts with any major gems). However, several participants including @ko1 (Koichi Sasada) were a little cautious about introducing the new terminology "create" into Ruby core, and matz understood that.

@matsuda (Akira Matsuda) and @mame (Yusuke Endoh) prefer Hash.new(capacity: 4096). This is a bit incompatible, but we searched gem-codesearch with the query '\bHash\.new\(\w+: ' and found less than 20 results (manually excluding Foo::Bar::Hash.new(...) which is perhaps different from ::Hash). Moreover, some of the results seemed to misunderstand Hash.new(foo: 1) as { foo: 1 }. (The most examples are rspec; maybe

because let(:option) { { foo: 1 } } looks bad, people inadvertently rewrote it with let(:option) { Hash.new(foo: 1) }.)

Therefore, how about deprecating giving the keyword to Hash.new and then introducing Hash.new(capacity: 4096)? @matz (Yukihiro Matsumoto) said this is also acceptable if the incompatibility is not a big problem.

(Off-topic: Array.new(capacity: 4096) is not yet available; I wonder if people want Hash.new(capacity: 4096) more than Array?)

#4 - 01/20/2023 08:42 AM - byroot (Jean Boussier)

Well, Hash.new(capacity: 4096) was definitely my first pick, so this is great news IMO.

> how about deprecating giving the keyword to Hash.new and then introducing Hash.new(capacity: 4096)?

What would be the timeline?

Deprecate in 3.3 and break in 3.4?

> Off-topic: Array.new(capacity: 4096) is not yet available; I wonder if people want Hash.new(capacity: 4096) more than Array?

I think it's in part because Array.new(4096) while not exactly the same, already somewhat works. I'd be happy to add Array.new(capacity: 4096) though, but it has a similar backward compatibility concern doesn't it?

#5 - 01/23/2023 02:47 AM - mame (Yusuke Endoh)

byroot (Jean Boussier) wrote in #note-4:

> What would be the timeline?
>
> Deprecate in 3.3 and break in 3.4?

That would be the fastest way.

> > Off-topic: Array.new(capacity: 4096) is not yet available; I wonder if people want Hash.new(capacity: 4096) more than Array?
>
> I think it's in part because Array.new(4096) while not exactly the same, already somewhat works. I'd be happy to add Array.new(capacity: 4096) though, but it has a similar backward compatibility concern doesn't it?

Fortunately, it raises an error: Array.new(capacity: 4096) #=> no implicit conversion of Hash into Integer (TypeError). So I don't see a big problem with changing this. Anyway, I think we need a separate ticket if we introduce it.

#6 - 01/24/2023 02:23 PM - Eregon (Benoit Daloze)

If we use Hash.new(capacity: 4096) to set the capacity, then Hash.new({ capacity: 4096 }) should keep the semantics of: { capacity: 4096 } is the default value of the new Hash.
I think it's a little bit hacky/unclear/source of confusion, but still I'm not against it.

#7 - 05/05/2023 11:59 PM - Dan0042 (Daniel DeLorme)

Previousy, a capacity reader/writer was suggested by @byroot (Jean Boussier) in #18683#note-2

I would like to see this idea considered more seriously because

1. It doesn't need to change anything to the initialize arguments of Array/Hash/String, which are already quite complex enough
2. The same API can be used for any class; it's nicely consistent and easy to remember
3. It's more versatile, as it can be used more than once after object creation, ex:

```
buffer = String.new #this example is with String, but the same could apply to Hash/Array
while line = gets
  #increase buffer capacity by chunks of 10k
  buffer.capacity += 10000 if buffer.capacity < buffer.bytesize + line.bytesize
  buffer << line
end
buffer.capacity = 0 #trim buffer to minimal size (aka "right-size")
buffer.capacity == buffer.bytesize #=> true
```

#8 - 05/06/2023 05:31 AM - ianks (Ian Ker-Seymer)

I worry that new Rubyists might be confused with the Hash.new(capacity: n) semantics.

For example, Hash[capacity: 5] can look very similar to Hash.new(capacity: 5). It wouldn't be unreasonable to assume they are the same thing… But

you'd be in for an unexpected surprise.

To me Hash.with_capacity clearly communicates what's happening. Anyone can understand it at first glance.

#9 - 05/06/2023 05:39 AM - byroot (Jean Boussier)

> For example, Hash[capacity: 5] can look very similar to Hash.new(capacity: 5).

That seems like a very handwavy argument to me. I really don't see how the two could possibly be confused.

#10 - 05/09/2023 12:57 PM - Dan0042 (Daniel DeLorme)

ianks (Ian Ker-Seymer) wrote in #note-8:

> To me Hash.with_capacity clearly communicates what's happening. Anyone can understand it at first glance.

Hash.with_capacity is not composable. What should you do if you want a default value/proc AND a capacity?

```
h = Hash.with_capacity(100)
h.default = default_value  #this?? a bit ugly imho
```

Hash#with_capacity would be better, then you could do Hash.new(default_value).with_capacity(400) similar to compare_by_identity usage.
But at that point it's imho better to have Hash.new(default_value).tap{ _1.capacity = 400 }
Or the best: Hash.new(default_value).tap{ .capacity = 400 } ;-)

#11 - 05/19/2023 10:04 AM - byroot (Jean Boussier)

This was discussed in the last dev meeting. The conclusion was:

> In 3.3 it throws error all keyword arguments to Hash.new. Then Ruby 3.4 allows that Hash.new will accept capacity keyword argument.

#12 - 05/23/2023 10:44 AM - byroot (Jean Boussier)

Correction:

> In 3.3 it throws error all keyword arguments to Hash.new.

Was a misunderstanding.

What was actually agreed was a deprecation warning, I modified the pull request accordingly.

#13 - 05/23/2023 01:51 PM - Anonymous

- Status changed from Open to Closed

Applied in changeset git|31ac8efca8ecb574e1e7b7c32cce54cb1b97f19a.

Hash.new: print a deprecation warning when receiving keyword arguments (#7828)

[Feature #19236]

In Ruby 3.3, Hash.new shall print a deprecation warning if keyword arguments
are passed instead of treating them as an implicit positional Hash.

This will allow to safely introduce a capacity keyword argument in 3.4

Co-authored-by: Jean Boussier byroot@ruby-lang.org

#14 - 05/23/2023 01:53 PM - byroot (Jean Boussier)

- Status changed from Closed to Open

- Target version deleted (3.3)

Reopening as the merged commit is the Ruby 3.3 part.

I'll implement the 3.4 next year.

#15 - 03/26/2024 07:20 AM - byroot (Jean Boussier)

Implemented Hash.new(capacity:) in https://github.com/ruby/ruby/pull/10357

#16 - 07/07/2024 01:41 AM - shan (Shannon Skipper)

I'm really looking forward to this feature being available via a Ruby interface.

#17 - 07/08/2024 10:24 AM - byroot (Jean Boussier)

- Status changed from Open to Closed

Applied in changeset git|9594db0cf28d7bc10bfc46142239191a11f1dbbe.

Implement Hash.new(capacity:)

[Feature #19236]

When building a large hash, pre-allocating it with enough capacity can save many re-hashes and significantly improve performance.

```
/opt/rubies/3.3.0/bin/ruby --disable=gems -rrubygems -I./benchmark/lib ./benchmark/benchmark-driver/exe/benchm
ark-driver \
                    --executables="compare-ruby::../miniruby-master -I.ext/common --disable-gem" \
                    --executables="built-ruby::./miniruby --disable-gem" \
                    --output=markdown --output-compare -v $(find ./benchmark -maxdepth 1 -name 'hash_new' -o -
name '*hash_new*.yml' -o -name '*hash_new*.rb' | sort)
compare-ruby: ruby 3.4.0dev (2024-03-25T11:48:11Z master f53209f023) +YJIT dev [arm64-darwin23]
last_commit=[ruby/irb] Cache RDoc::RI::Driver.new (https://github.com/ruby/irb/pull/911)
built-ruby: ruby 3.4.0dev (2024-03-25T15:29:40Z hash-new-rb 77652b08a2) +YJIT dev [arm64-darwin23]
warming up...
```

	compare-ruby	built-ruby
new	7.614M	5.976M
	1.27x	-
new_with_capa_1k	13.931k	15.698k
	-	1.13x
new_with_capa_100k	124.746	148.283
	-	1.19x