Description

Behavior

If passed freeze: true, all the deserialized objects should be frozen, and if possible, strings should be deduped.

This is similar to the freeze option recently added to JSON (https://github.com/flori/json/pull/447), Psych (https://github.com/ruby/psych/pull/414) and MessagePack (https://github.com/msgpack/msgpack-ruby/pull/194).

Use cases

This option is useful in many scenarios:

- If the deserialized data is meant to stay on the heap for the lifetime of the program, the string deduplication reduce the memory overhead, and all objects being frozen improve copy on write and ensure that static data isn't accidentally mutated.
- If the deserialized data is used in a memory cache or similar, deep freezing it protect against mutation and allow to return the value directly without first deep cloning it.
- While not very performant, it can be used as a deep_freeze mechanism with Marshal.load(Marshal.dump(object), freeze: true).

Snippets

```ruby
payload = Marshal.dump({"foo" => ["bar"]})
object = Marshal.load(payload, freeze: true)

object.frozen?
object.dig("foo").frozen?
object.dig("foo", 1).frozen?

Marshal.load(payload, ->(obj) { raise "unexpected" unless obj.frozen? }, freeze: true)

def cache_get(key)
  if entry = in_memory_cache.get(key)
    return entry
  end

  if payload = network_cache.get(key)
    object = Marshal.load(payload, freeze: true)
    in_memory_cache.set(key, object)
  # if the object tree wasn't frozen, we'd need to deep dup to avoid mutation.
  object
  end
end
```

Associated revisions

Revision afcbb501 - 10/05/2021 04:34 PM - byroot (Jean Boussier)

marshal.c Marshal.load accepts a freeze: true option.

Fixes [Feature #18148]

When set, all the loaded objects are returned as frozen.

If a proc is provided, it is called with the objects already frozen.
It would be helpful for us if you could add a simple code example for a feature proposal.

dump = Marshal.dump("foo", "bar", "baz")
ary = Marshal.load(dump, freeze: true)
p ary #=> ["foo", "bar", "baz"]
p ary.frozen? #=> true
p ary[0].frozen? #=> true

Marshal.load() accepts proc which can manipulate the loaded object:

str = Marshal.dump("a", 1.0 ** 10, 1.0, :foo)
p Marshal.load(str, proc { |obj| [obj] }) #=> [["a"], [1], [10.0], [:foo]]

When freeze:true is specified, only returned value from proc will be frozen?

It could be either really, but I think the proc should be called with the objects already frozen, mostly to allow deserializing strings with rb_interned_str...

It would be helpful for us if you could add a simple code example for a feature proposal.

Sure I'll add some.

Could you please create a patch for your proposal?

That was my intent, it might take me a couple days though.

I implemented a patch https://github.com/ruby/ruby/pull/4859

NB: it does include some extra fixes for https://bugs.ruby-lang.org/issues/18141#change-93742, because the implementation wouldn't have been
possible without.

---

#10 - 10/05/2021 04:31 PM - byroot (Jean Boussier)
- Status changed from Assigned to Closed

Applied in changeset git|afcbb501ac17ba2ad5370ada5fd26e8dda9a5aaa.

marshal.c Marshal.load accepts a freeze: true option. 

Fixes [Feature #18148]

When set, all the loaded objects are returned as frozen.

If a proc is provided, it is called with the objects already frozen.