Ruby trunk - Feature #6962

Use lighter hash structure for methods table, instance variable positions, constants

09/02/2012 09:59 PM - funny_falcon (Yura Sokolov)

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

Description

I've experimented with replacing struct st_table with lighter hash structure.

I were inspired by Lua tables, so that hash were created similar:

- open addressing but with elements chaining instead of any kind of probing
- there is sibling probing on insert, to ensure cache locality
- on insert, already inserted collision item which is not in main position is moved to free place, so that chains started from different main positions are not joined. This allows to keep chains short.
- struct header is 24 bytes on 64bit platform (compared with 40 bytes for st_table), so that i've placed them directly into rb_classext_struct instead of separate allocation.

Since keys are integers (ID), there is no separate field for hash. And I used unsigned int instead of
unsigned long for key, cause ID almost never reach 2**32. It allows to pack whole entry into 16 bytes
on 64bit platform (compared to 48 bytes for st_table_entry).

Such structure gives about 6% improvement without GC tuning, and 10% improvement when GC is tuned
(I've used RUBY_GC_MALLOC_LIMIT=60000000 RUBY_FREE_MIN=200000)

Also I add per class method cache instead of single global hash array. It has advantage over global
cache cause there is no collision over different classes and methods. But it seems that Rails applications
too often creates singleton classes, so that improvement is not very big (but no slowdown too).

Currently I have patch for Ruby 1.9.3 https://github.com/funny-falcon/ruby/compare/ruby_1_9_3...sparse_array/ruby_1_9_3

Single incompatible external api change is almost neglible, cause rb_generic_ivar_table methos is not very usefull
https://github.com/funny-falcon/ruby/compare/ruby_1_9_3...sparse_array/ruby_1_9_3#L5L852

Silly, but there is too many changes in trunk, so that, I could not simply rebase it, but ought to make whole work again.
I'll try to do it shortly.

With regards, Sokolov Yura aka funny-falcon.

Related issues:

Related to Ruby trunk - Feature #11420: Introduce ID key table into MRI	**Closed**

History

#1 - 09/02/2012 10:27 PM - funny_falcon (Yura Sokolov)

It seems that I'm using "Coalesced Hashing" http://en.wikipedia.org/wiki/Coalesced_hashing
with some improvements:

- moving collision items which is not in its main position (as in Lua tables)
- sibling probing for free items (test 4 closest items before using tablewise pointer to free position)

#2 - 11/20/2012 11:27 PM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee changed from matz (Yukihiro Matsumoto) to ko1 (Koichi Sasada)

- Target version set to 2.6

ko1 or nobu, please check this (towards next minor).

--
Yusuke Endoh mame@tsg.ne.jp

#3 - 11/22/2012 06:21 PM - funny_falcon (Yura Sokolov)

I've updated patch to latest ruby_1_9_3.
Url is the same:
https://github.com/funny-falcon/ruby/compare/ruby_1_9_3...sparse_array/ruby_1_9_3

#4 - 08/06/2015 08:56 AM - ko1 (Koichi Sasada)

- Related to Feature #11420: Introduce ID key table into MRI added

#5 - 08/21/2015 11:52 PM - ko1 (Koichi Sasada)

- Status changed from Assigned to Closed

Continue on Feature #11420.