This reduces memory overhead of ivars for common types such as T_DATA the same way T_OBJECT does it.

For 9992 accepted clients on an OpenSSL server, this reduces RSS memory from 77160K to 69248K on x86-64 with the attached ssl.rb script. Connecting client process was reduced from 246312K to 230724K RSS.

OpenSSL 1.0.1e-2+deb7u16 on Debian 7

Associated revisions
Revision 9d9aea7f - 05/29/2015 11:42 PM - normal
variable.c: use indices for generic ivars

This reduces memory overhead of ivars for common types such as T_DATA the same way T_OBJECT does it.

For 9992 accepted clients on an OpenSSL server, this reduces memory from 77160K to 69248K with the script in https://bugs.ruby-lang.org/issues/11170


git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@50678 b2dd03c8-39d4-4d8f-98ff-823fe69b08e

Revision 50678 - 05/29/2015 11:42 PM - normalperson (Eric Wong)
variable.c: use indices for generic ivars

This reduces memory overhead of ivars for common types such as T_DATA the same way T_OBJECT does it.

For 9992 accepted clients on an OpenSSL server, this reduces memory from 77160K to 69248K with the script in https://bugs.ruby-lang.org/issues/11170


Revision 50678 - 05/29/2015 11:42 PM - normal
This reduces memory overhead of ivars for common types such as T_DATA the same way T_OBJECT does it.

For 9992 accepted clients on an OpenSSL server, this reduces memory from 77160K to 69248K with the script in https://bugs.ruby-lang.org/issues/11170


**Revision 50678 - 05/29/2015 11:42 PM - normal**

variable.c: use indices for generic ivars

This reduces memory overhead of ivars for common types such as T_DATA the same way T_OBJECT does it.

For 9992 accepted clients on an OpenSSL server, this reduces memory from 77160K to 69248K with the script in https://bugs.ruby-lang.org/issues/11170


**Revision 50678 - 05/29/2015 11:42 PM - normal**

variable.c: use indices for generic ivars

This reduces memory overhead of ivars for common types such as T_DATA the same way T_OBJECT does it.

For 9992 accepted clients on an OpenSSL server, this reduces memory from 77160K to 69248K with the script in https://bugs.ruby-lang.org/issues/11170


**Revision 50678 - 05/29/2015 11:42 PM - normal**

variable.c: use indices for generic ivars

This reduces memory overhead of ivars for common types such as T_DATA the same way T_OBJECT does it.

For 9992 accepted clients on an OpenSSL server, this reduces memory from 77160K to 69248K with the script in https://bugs.ruby-lang.org/issues/11170


**Revision 6cd5825 - 05/30/2015 12:20 AM - normal**

variable.c: avoid compatibility table with generic ivars

This recovers and improves performance of Marshal.dump/load on Time objects compared to when we implemented generic ivars entirely using st_table.
This also recovers some performance on other generic ivar objects, but does not bring Marshal.dump/load performance up to previous speeds.

benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name    trunk   geniv   after
marshal_dump_flo        0.343   0.334   0.335
marshal_dump_load_geniv 0.487   0.527   0.495
marshal_dump_load_time  1.262   1.401   1.257

Speedup ratio: compare with the result of 'trunk' (greater is better)
name    geniv   after
marshal_dump_flo        1.026   1.023
marshal_dump_load_geniv 0.925   0.985
marshal_dump_load_time  0.901   1.004

- include/ruby/intern.h (rb_generic_ivar_table): deprecate
- internal.h (rb_attr_delete): declare
- marshal.c (has_ivars): use rb_ivar_foreach (w_iivar): ditto (w_object): update for new interface
- time.c (time_mload): use rb_attr_delete
- variable.c (generic_ivar_delete): implement (rb_ivar_delete): ditto (rb_attr_delete): ditto

Revision 50680 - 05/30/2015 12:20 AM - normalperson (Eric Wong)
variable.c: avoid compatibility table with generic ivars

This recovers and improves performance of Marshal.dump/load on Time objects compared to when we implemented generic ivars entirely using st_table.

This also recovers some performance on other generic ivar objects, but does not bring Marshal.dump/load performance up to previous speeds.

benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name    trunk   geniv   after
marshal_dump_flo        0.343   0.334   0.335
marshal_dump_load_geniv 0.487   0.527   0.495
marshal_dump_load_time  1.262   1.401   1.257

Speedup ratio: compare with the result of 'trunk' (greater is better)
name    geniv   after
marshal_dump_flo        1.026   1.023
marshal_dump_load_geniv 0.925   0.985
marshal_dump_load_time  0.901   1.004

- include/ruby/intern.h (rb_generic_ivar_table): deprecate
- internal.h (rb_attr_delete): declare
- marshal.c (has_ivars): use rb_ivar_foreach (w_iivar): ditto (w_object): update for new interface
- time.c (time_mload): use rb_attr_delete
- variable.c (generic_ivar_delete): implement (rb_ivar_delete): ditto (rb_attr_delete): ditto

Revision 50680 - 05/30/2015 12:20 AM - normal
variable.c: avoid compatibility table with generic ivars

This recovers and improves performance of Marshal.dump/load on Time objects compared to when we implemented generic ivars entirely using st_table.

This also recovers some performance on other generic ivar objects, but does not bring Marshal.dump/load performance up to previous speeds.

benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name    trunk   geniv   after
marshal_dump_flo        0.343   0.334   0.335
include/ruby/intern.h (rb_generic_ivar_table): deprecate
internal.h (rb_attr_delete): declare
marshal.c (has_ivars): use rb_ivar_foreach (w_ivar): ditto (w_object): update for new interface
time.c (time_mload): use rb_attr_delete
variable.c (generic_ivar_delete): implement (rb_ivar_delete): ditto (rb_attr_delete): ditto [ruby-core:69323] [Feature #11170]

Revision 50680 - 05/30/2015 12:20 AM - normal

variable.c: avoid compatibility table with generic ivars

This recovers and improves performance of Marshal.dump/load on Time objects compared to when we implemented generic ivars entirely using st_table.

This also recovers some performance on other generic ivar objects, but does not bring bring Marshal.dump/load performance up to previous speeds.

benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name trunk geniv after
marshal_dump_flo 0.343 0.334 0.335
marshal_dump_load_geniv 0.487 0.527 0.495
marshal_dump_load_time 1.262 1.401 1.257

Speedup ratio: compare with the result of 'trunk' (greater is better)
name geniv after
marshal_dump_flo 1.026 1.023
marshal_dump_load_geniv 0.925 0.985
marshal_dump_load_time 0.901 1.004

• include/ruby/intern.h (rb_generic_ivar_table): deprecate
• internal.h (rb_attr_delete): declare
• marshal.c (has_ivars): use rb_ivar_foreach (w_ivar): ditto (w_object): update for new interface
• time.c (time_mload): use rb_attr_delete
• variable.c (generic_ivar_delete): implement (rb_ivar_delete): ditto (rb_attr_delete): ditto [ruby-core:69323] [Feature #11170]

Revision 50680 - 05/30/2015 12:20 AM - normal

variable.c: avoid compatibility table with generic ivars

This recovers and improves performance of Marshal.dump/load on Time objects compared to when we implemented generic ivars entirely using st_table.

This also recovers some performance on other generic ivar objects, but does not bring bring Marshal.dump/load performance up to previous speeds.

benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name trunk geniv after
marshal_dump_flo 0.343 0.334 0.335
marshal_dump_load_geniv 0.487 0.527 0.495
marshal_dump_load_time 1.262 1.401 1.257

Speedup ratio: compare with the result of 'trunk' (greater is better)
name geniv after
marshal_dump_flo 1.026 1.023
marshal_dump_load_geniv 0.925 0.985
marshal_dump_load_time 0.901 1.004

• include/ruby/intern.h (rb_generic_ivar_table): deprecate
• internal.h (rb_attr_delete): declare
Revision 50680 - 05/30/2015 12:20 AM - normal

variable.c: avoid compatibility table with generic ivars

This recovers and improves performance of Marshal.dump/load on
Time objects compared to when we implemented generic ivars
entirely using st_table.

This also recovers some performance on other generic ivar objects,
but does not bring bring Marshal.dump/load performance up to
previous speeds.

benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name    trunk   geniv   after
marshal_dump_flo  0.343   0.334   0.335
marshal_dump_load_geniv 0.487   0.527   0.495
marshal_dump_load_time  1.262   1.401   1.257

Speedup ratio: compare with the result of 'trunk' (greater is better)
name    geniv   after
marshal_dump_flo  1.026   1.023
marshal_dump_load_geniv  0.925   0.985
marshal_dump_load_time   0.901   1.004

include/ruby/intern.h (rb_generic_ivar_table): deprecate
internal.h (rb_attr_delete): declare
marshal.c (has_ivars): use rb_ivar_foreach (w_ivar): ditto (w_object): update for new interface
time.c (time_mload): use rb_attr_delete
variable.c (generic_ivar_delete): implement (rb_ivar_delete): ditto (rb_attr_delete): ditto [ruby-core:69323] [Feature #11170]
variable.c: use indices for generic ivars

This reduces memory overhead of ivars for common types such as T_DATA the same way T_OBJECT does it.

For 9992 accepted clients on an OpenSSL server, this reduces memory from 77160K to 69248K with the script in https://bugs.ruby-lang.org/issues/11170