Ruby - Feature #10575

[RFC] struct: avoid all O(n) behavior on access

12/06/2014 12:31 AM - normalperson (Eric Wong)

Status:	Closed
Priority:	Normal
Assignee:	ko1 (Koichi Sasada)
Target version:	2.2.0

Description

This avoids O(n) on lookups with structs over 10 members.
This also avoids O(n) behavior on all assignments on Struct members.
Members 0..9 still use existing C methods to read in O(1) time

Benchmark results:

```
vm2_struct_big_aref_hi* 1.305
vm2_struct_big_aref_lo* 1.157
vm2_struct_big_aset* 3.306
vm2_struct_small_aref* 1.015
vm2_struct_small_aset* 3.273
```

Note: I chose use loading instructions from an array instead of writing
directly to linked-lists in compile.c for ease-of-maintainability. We
may move the method definitions to prelude.rb-like files in the future.

I have also tested this patch with the following patch to disable
the C ref_func methods and ensured the test suite and rubyspec works

```
--- a/struct.c
+++ b/struct.c
@@ -209,7 +209,7 @@ setup_struct(VALUE nstr, VALUE members)
 ID id = SYM2ID(ptr_members[i]);
 VALUE off = LONG2NUM(i);

- if (i < N_REF_FUNC) {
+ if (0 && i < N_REF_FUNC) {
     rb_define_method_id(nstr, id, ref_func[i], 0);
 }
 else {
```

Associated revisions

Revision 65651b34b1b5c2ef556ed44d5ebbd98838cfe8e6 - 12/09/2014 03:43 PM - Eric Wong

struct: avoid all O(n) behavior on access

This avoids O(n) on lookups with structs over 10 members.
This also avoids O(n) behavior on all assignments on Struct members.
Members 0..9 still use existing C methods to read in O(1) time

Benchmark results:

vm2_struct_big_aref_hi* 1.305
vm2_struct_big_aref_lo* 1.157
vm2_struct_big_aset* 3.306
vm2_struct_small_aref* 1.015
vm2_struct_small_aset* 3.273

Note: I chose use loading instructions from an array instead of writing
directly to linked-lists in compile.c for ease-of-maintainability. We
may move the method definitions to prelude.rb-like files in the future.

I have also tested this patch with the following patch to disable
the C ref_func methods and ensured the test suite and rubyspec works

```
    --- a/struct.c
```

```
    +++ b/struct.c
    @@ -209,7 +209,7 @@ setup_struct(VALUE nstr, VALUE members)
             ID id = SYM2ID(ptr_members[i]);
             VALUE off = LONG2NUM(i);

    -        if (i < N_REF_FUNC) {
    +        if (0 && i < N_REF_FUNC) {
                 rb_define_method_id(nstr, id, ref_func[i], 0);
             }
             else {
```

- iseq.c (rb_method_for_self_aref, rb_method_for_self_aset):
  new methods to generate bytecode for struct.c
  [Feature #10575]
- struct.c (rb_struct_ref, rb_struct_set): remove
  (define_aref_method, define_aset_method): new functions
  (setup_struct): use new functions
- test/ruby/test_struct.rb: add test for struct >10 members
- benchmark/bm_vm2_struct_big_aref_hi.rb: new benchmark
- benchmark/bm_vm2_struct_big_aref_lo.rb: ditto
- benchmark/bm_vm2_struct_big_aset.rb: ditto
- benchmark/bm_vm2_struct_small_aref.rb: ditto
- benchmark/bm_vm2_struct_small_aset.rb: ditto

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@48748 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 65651b34 - 12/09/2014 03:43 PM - Eric Wong

struct: avoid all O(n) behavior on access

This avoids O(n) on lookups with structs over 10 members.
This also avoids O(n) behavior on all assignments on Struct members.
Members 0..9 still use existing C methods to read in O(1) time

Benchmark results:

vm2_struct_big_aref_hi* 1.305
vm2_struct_big_aref_lo* 1.157
vm2_struct_big_aset* 3.306
vm2_struct_small_aref* 1.015
vm2_struct_small_aset* 3.273

Note: I chose use loading instructions from an array instead of writing
directly to linked-lists in compile.c for ease-of-maintainability. We
may move the method definitions to prelude.rb-like files in the future.

I have also tested this patch with the following patch to disable
the C ref_func methods and ensured the test suite and rubyspec works

```
    --- a/struct.c
    +++ b/struct.c
    @@ -209,7 +209,7 @@ setup_struct(VALUE nstr, VALUE members)
             ID id = SYM2ID(ptr_members[i]);
             VALUE off = LONG2NUM(i);

    -        if (i < N_REF_FUNC) {
    +        if (0 && i < N_REF_FUNC) {
                 rb_define_method_id(nstr, id, ref_func[i], 0);
             }
             else {
```

- iseq.c (rb_method_for_self_aref, rb_method_for_self_aset):
  new methods to generate bytecode for struct.c
  [Feature #10575]
- struct.c (rb_struct_ref, rb_struct_set): remove
  (define_aref_method, define_aset_method): new functions
  (setup_struct): use new functions
- test/ruby/test_struct.rb: add test for struct >10 members
- benchmark/bm_vm2_struct_big_aref_hi.rb: new benchmark
- benchmark/bm_vm2_struct_big_aref_lo.rb: ditto
- benchmark/bm_vm2_struct_big_aset.rb: ditto
- benchmark/bm_vm2_struct_small_aref.rb: ditto
- benchmark/bm_vm2_struct_small_aset.rb: ditto

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@48748 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 8efe878d11a136c8ea209eaf1448ee6a788c4814 - 12/10/2014 04:39 AM - nobu (Nobuyoshi Nakada)

struct.c: use iseqval

- struct.c (define_aref_method, define_aset_method): use iseq
  VALUE instead of rb_iseq_t to prevent from GC, as RB_GC_GUARD
  makes sense only for local variables. [Feature #10575]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@48754 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 8efe878d - 12/10/2014 04:39 AM - nobu (Nobuyoshi Nakada)

struct.c: use iseqval

- struct.c (define_aref_method, define_aset_method): use iseq
  VALUE instead of rb_iseq_t to prevent from GC, as RB_GC_GUARD
  makes sense only for local variables. [Feature #10575]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@48754 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 12/09/2014 08:52 AM - nobu (Nobuyoshi Nakada)

- Description updated

Looks fine.

#2 - 12/09/2014 03:44 PM - Anonymous

- Status changed from Open to Closed

- % Done changed from 0 to 100

Applied in changeset r48748.

---

struct: avoid all O(n) behavior on access

This avoids O(n) on lookups with structs over 10 members.
This also avoids O(n) behavior on all assignments on Struct members.
Members 0..9 still use existing C methods to read in O(1) time

Benchmark results:

vm2_struct_big_aref_hi* 1.305
vm2_struct_big_aref_lo* 1.157
vm2_struct_big_aset* 3.306
vm2_struct_small_aref* 1.015
vm2_struct_small_aset* 3.273

Note: I chose use loading instructions from an array instead of writing
directly to linked-lists in compile.c for ease-of-maintainability. We
may move the method definitions to prelude.rb-like files in the future.

I have also tested this patch with the following patch to disable
the C ref_func methods and ensured the test suite and rubyspec works

```
--- a/struct.c
+++ b/struct.c
@@ -209,7 +209,7 @@ setup_struct(VALUE nstr, VALUE members)
 ID id = SYM2ID(ptr_members[i]);
 VALUE off = LONG2NUM(i);

- if (i < N_REF_FUNC) {
+ if (0 && i < N_REF_FUNC) {
     rb_define_method_id(nstr, id, ref_func[i], 0);
 }
 else {
```

- iseq.c (rb_method_for_self_aref, rb_method_for_self_aset):
  new methods to generate bytecode for struct.c
  [Feature #10575]
- struct.c (rb_struct_ref, rb_struct_set): remove
  (define_aref_method, define_aset_method): new functions

(setup_struct): use new functions
- test/ruby/test_struct.rb: add test for struct >10 members
- benchmark/bm_vm2_struct_big_aref_hi.rb: new benchmark
- benchmark/bm_vm2_struct_big_aref_lo.rb: ditto
- benchmark/bm_vm2_struct_big_aset.rb: ditto
- benchmark/bm_vm2_struct_small_aref.rb: ditto
- benchmark/bm_vm2_struct_small_aset.rb: ditto

#3 - 12/10/2014 04:47 PM - funny_falcon (Yura Sokolov)

Couple of other struct optimizations in #10585

#4 - 12/10/2014 06:46 PM - ko1 (Koichi Sasada)

Sorry for late.

> Benchmark results:
>
> vm2_struct_big_aref_hi* 1.305
> ...

how to read? seconds? compared ratio with non-opt version?

#5 - 12/11/2014 09:38 PM - normalperson (Eric Wong)

speedup ratio (from benchmark/driver.rb)

#6 - 12/11/2014 10:28 PM - normalperson (Eric Wong)

funny.falcon@gmail.com wrote:

> Couple of other struct optimizations in #10585

Thanks. Btw, can you reproduce the issue in [ruby-core:66762]? I cannot reproduce it anymore on 32-bit x86, so I'm not sure if further digging is required [ruby-core:66764]

Files

struct_iseq-v1-r48725.patch	10.8 KB	12/06/2014	normalperson (Eric Wong)