Ruby - Misc #10278

[RFC] st.c: use ccan linked list

09/22/2014 05:32 AM - normalperson (Eric Wong)

Status:	Closed
Priority:	Normal
Assignee:	normalperson (Eric Wong)

Description

Mainly posting this for documentation purposes because it seems like
an obvious thing to try given we have ccan/list nowadays.

Having shorter code along branchless insert/delete, and using a common
linked-list API is very appealing.

On the other hand, benchmark results are a mixed bag:

```
http://80x24.org/bmlog-20140922-032221.13002
```

Also, I may have introduced new bugs the tests didn't catch.
The st_foreach* functions get a bit strange when dealing with
packed-to-unpacked transitions while iterating.

Great thing: bighash is faster (as expected) because of branchless
linked list insertion. However, the major speedup in bighash probably
isn't too important, most hashes are small and users never notice.

```
vm2_bighash* 1.222
```

Also, we could introduce rb_hash_new_with_size() for use insns.def
(newhash) if people really care about the static bighash case (I don't
think many do).

Real regressions, iteration seems more complex because loop conditions
are more complex :<

```
hash_keys 0.978
hash_values 0.941
```

However, hash_keys/values regressions are pretty small.

Things that worry me:

```
vm1_attr_ivar* 0.736
vm1_attr_ivar_set* 0.845
```

WTF? I reran the attr_ivar tests, and the numbers got slightly better:

```
 ["vm1_attr_ivar",
  [[1.851297842,
    1.549076322,
    1.623306027,
    1.956916541,
    1.533218607,
    1.554089054,
    1.702590516,
    1.789863782,
    1.711815018,
    1.851260599],
   [1.825423191,
    1.824934062,
    1.542471471,
    1.868502091,
    1.79106375,
    1.884568825,
    1.850712387,
    1.797538962,
    2.165696827,
    1.866671482]]],
 ["vm1_attr_ivar_set",
  [[1.926496052,
    2.04742421,
    2.025571131,
    2.047656291,
    2.043747069,
    2.099586827,
    1.953769267,
    2.017580504,
    2.440432603,
    2.111254634],
   [2.365839125,
    2.076282818,
    2.112784977,
    2.118754445,
    2.091752673,
    2.161164561,
    2.107439445,
    2.128147747,
    2.945295069,
    2.131679632]]]]

Elapsed time: 91.963235593 (sec)
-----------------------------------------------------------
benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name orig stll
loop_whileloop 0.672 0.670
vm1_attr_ivar* 0.861 0.872
vm1_attr_ivar_set* 1.255 1.406

Speedup ratio: compare with the result of `orig' (greater is better)
name stll
loop_whileloop 1.002
vm1_attr_ivar* 0.987
vm1_attr_ivar_set* 0.892
```

Note: these tests do not even hit st, and even if they did, these are
tiny tables which are packed so the linked-list implementation has no
impact (especially not on lookup tests)

So yeah, probably something messy with the CPU caches.
I always benchmark with the performance CPU governor, and the
rerun ivar numbers are run with CPU pinned to a single core.
CPU: AMD FX-8320 Maybe I can access my other systems later.

Related issues:

Related to Ruby - Feature #10321: [PATCH] test st_foreach{,_check} for packed...	**Closed**	**10/03/2014**

Associated revisions

Revision d8748874cca191f2244595d25f95b091dd423150 - 06/25/2015 07:01 PM - Eric Wong

st.c: use ccan linked-list

This improves the bm_vm2_bighash benchmark significantly by
removing branches during insert, but slows down anything
requiring iteration with the more complex loop termination
checking.

Speedup ratio of 1.10 - 1.20 is typical for the vm2_bighash
benchmark.

- include/ruby/st.h (struct st_table): hide struct list_head
- st.c (struct st_table_entry): adjust struct
  (head, tail): remove shortcut macros
  (st_head): new wrapper function
  (st_init_table_with_size): adjust to new struct and API
  (st_clear): ditto
  (add_direct): ditto
  (unpack_entries): ditto
  (rehash): ditto
  (st_copy): ditto
  (remove_entry): ditto
  (st_shift): ditto
  (st_foreach_check): ditto
  (st_foreach): ditto
  (get_keys): ditto
  (get_values): ditto
  (st_values_check): ditto
  (st_reverse_foreach_check): ditto (unused)
  (st_reverse_foreach): ditto (unused)
  [ruby-core:69726] [Misc #10278]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@51034 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision d8748874 - 06/25/2015 07:01 PM - Eric Wong

st.c: use ccan linked-list

This improves the bm_vm2_bighash benchmark significantly by
removing branches during insert, but slows down anything
requiring iteration with the more complex loop termination
checking.

Speedup ratio of 1.10 - 1.20 is typical for the vm2_bighash
benchmark.

- include/ruby/st.h (struct st_table): hide struct list_head
- st.c (struct st_table_entry): adjust struct
  (head, tail): remove shortcut macros
  (st_head): new wrapper function
  (st_init_table_with_size): adjust to new struct and API
  (st_clear): ditto
  (add_direct): ditto
  (unpack_entries): ditto
  (rehash): ditto
  (st_copy): ditto
  (remove_entry): ditto
  (st_shift): ditto
  (st_foreach_check): ditto
  (st_foreach): ditto
  (get_keys): ditto
  (get_values): ditto
  (st_values_check): ditto
  (st_reverse_foreach_check): ditto (unused)
  (st_reverse_foreach): ditto (unused)
  [ruby-core:69726] [Misc #10278]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@51034 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision d3725a86dee7c80dba1e16110545ff69c4ad9e1d - 06/26/2015 10:32 PM - Eric Wong

st.c: use ccan linked-list (try 2)

This improves the bm_vm2_bighash benchmark significantly by
removing branches during insert, but slows down anything
requiring iteration with the more complex loop termination
checking.

Speedup ratio of 1.10 - 1.20 is typical for the vm2_bighash
benchmark.

- include/ruby/st.h (struct st_table): hide struct list_head
- st.c (struct st_table_entry): adjust struct
  (head, tail): remove shortcut macros
  (st_head): new wrapper function
  (st_init_table_with_size): adjust to new struct and API

(st_clear): ditto
(add_direct): ditto
(unpack_entries): ditto
(rehash): ditto
(st_copy): ditto
(remove_entry): ditto
(st_shift): ditto
(st_foreach_check): ditto
(st_foreach): ditto
(get_keys): ditto
(get_values): ditto
(st_values_check): ditto
(st_reverse_foreach_check): ditto (unused)
(st_reverse_foreach): ditto (unused)
[ruby-core:69726] [Misc #10278]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@51044 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision d3725a86 - 06/26/2015 10:32 PM - Eric Wong

st.c: use ccan linked-list (try 2)

This improves the bm_vm2_bighash benchmark significantly by
removing branches during insert, but slows down anything
requiring iteration with the more complex loop termination
checking.

Speedup ratio of 1.10 - 1.20 is typical for the vm2_bighash
benchmark.

- include/ruby/st.h (struct st_table): hide struct list_head
- st.c (struct st_table_entry): adjust struct
  (head, tail): remove shortcut macros
  (st_head): new wrapper function
  (st_init_table_with_size): adjust to new struct and API
  (st_clear): ditto
  (add_direct): ditto
  (unpack_entries): ditto
  (rehash): ditto
  (st_copy): ditto
  (remove_entry): ditto
  (st_shift): ditto
  (st_foreach_check): ditto
  (st_foreach): ditto
  (get_keys): ditto
  (get_values): ditto
  (st_values_check): ditto
  (st_reverse_foreach_check): ditto (unused)
  (st_reverse_foreach): ditto (unused)
  [ruby-core:69726] [Misc #10278]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@51044 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision c31b0def42942c7d9f61b87e9aedf665363970ae - 06/29/2015 06:10 PM - Eric Wong

st.c: use ccan linked-list (try 3)

This improves the bm_vm2_bighash benchmark significantly by
removing branches during insert, but slows down anything
requiring iteration with the more complex loop termination
checking.

Speedup ratio of 1.10 - 1.20 is typical for the vm2_bighash
benchmark.

v3 - st_head calculates list_head address in two steps
to avoid a bug in old gcc 4.4 (Debian 4.4.7-2)
bug which incorrectly warned with:
warning: dereferencing pointer '({anonymous})' does break
strict-aliasing rules

- include/ruby/st.h (struct st_table): hide struct list_head
- st.c (struct st_table_entry): adjust struct
  (head, tail): remove shortcut macros
  (st_head): new wrapper function

(st_init_table_with_size): adjust to new struct and API
(st_clear): ditto
(add_direct): ditto
(unpack_entries): ditto
(rehash): ditto
(st_copy): ditto
(remove_entry): ditto
(st_shift): ditto
(st_foreach_check): ditto
(st_foreach): ditto
(get_keys): ditto
(get_values): ditto
(st_values_check): ditto
(st_reverse_foreach_check): ditto (unused)
(st_reverse_foreach): ditto (unused)
[ruby-core:69726] [Misc #10278]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@51064 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision c31b0def - 06/29/2015 06:10 PM - Eric Wong

st.c: use ccan linked-list (try 3)

This improves the bm_vm2_bighash benchmark significantly by
removing branches during insert, but slows down anything
requiring iteration with the more complex loop termination
checking.

Speedup ratio of 1.10 - 1.20 is typical for the vm2_bighash
benchmark.

v3 - st_head calculates list_head address in two steps
to avoid a bug in old gcc 4.4 (Debian 4.4.7-2)
bug which incorrectly warned with:
warning: dereferencing pointer '({anonymous})' does break
strict-aliasing rules

- include/ruby/st.h (struct st_table): hide struct list_head
- st.c (struct st_table_entry): adjust struct
  (head, tail): remove shortcut macros
  (st_head): new wrapper function
  (st_init_table_with_size): adjust to new struct and API
  (st_clear): ditto
  (add_direct): ditto
  (unpack_entries): ditto
  (rehash): ditto
  (st_copy): ditto
  (remove_entry): ditto
  (st_shift): ditto
  (st_foreach_check): ditto
  (st_foreach): ditto
  (get_keys): ditto
  (get_values): ditto
  (st_values_check): ditto
  (st_reverse_foreach_check): ditto (unused)
  (st_reverse_foreach): ditto (unused)
  [ruby-core:69726] [Misc #10278]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@51064 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 09/22/2014 06:34 AM - nobu (Nobuyoshi Nakada)

- Description updated

Probably, we should remove back member.

#2 - 09/22/2014 07:12 AM - normalperson (Eric Wong)

nobu@ruby-lang.org wrote:

> Probably, we should remove back member.

Just back and making it a singly-linked list?
st_delete would become O(n), unfortunately.

I also do not think going from 48 to 40 bytes makes a difference on most
x86-64 mallocs because (IIRC) the ABI requires 16-byte alignment.

If we can go from 48 => 32 bytes, great!
But I don't see what else to remove while keeping compatibility/speed :<

#3 - 09/22/2014 07:23 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Assigned

Indeed.

#4 - 09/22/2014 06:58 PM - normalperson (Eric Wong)

Better (at least more explainable) results on the Xeon:
http://80x24.org/spew/m/st-ccan-list-bench@meltdown.html

Will test on the old Phenom II, too.

#5 - 09/22/2014 11:40 PM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> Will test on the old Phenom II, too.

bighash looks nice as expected, haven't had time to look more into
these numbers but it's more consistent than the Vishera (FX-8320):
http://80x24.org/spew/m/20140922231823.GA21644%40dcvr.yhbt.net.html

#6 - 10/02/2014 06:58 PM - normalperson (Eric Wong)

A fixup patch for packed => unpacked transitions:

http://80x24.org/spew/m/st-ccan-ll-fixup-1%4080x24.org.txt

This needs tests, but it seems the packed => unpacked transitions during
iteration are totally untested in the current Ruby implementation.
Fortunately, it seems hash.c bans such transitions.

I suppose I can write tests to explicitly test for these changes,
but it may be easier and cheaper to bail out (possibly raise an error)

#7 - 10/03/2014 10:53 PM - normalperson (Eric Wong)

- Related to Feature #10321: [PATCH] test st_foreach{,_check} for packed-to-unpack change added

#8 - 10/04/2014 02:12 AM - normalperson (Eric Wong)

I like my original patch a little more, now, especially since it passes
the test in #10321. I'll squash the following simplfication on top
if I commit: http://80x24.org/spew/m/st-ll-foreach-simple%40whir.txt

#9 - 10/05/2014 12:49 PM - normalperson (Eric Wong)

Since we'll need it for st_reverse_foreach_check ([ruby-core:65408]),
I've implemented list_for_each_rev_safe to ccan/list:
https://lists.ozlabs.org/pipermail/ccan/2014-October/thread.html
It applies on top of two of my others intended for compile.c:
https://lists.ozlabs.org/pipermail/ccan/2014-September/thread.html

Also, updated bench results from the weird FX-8320 CPU after
simplifying the foreach loops a little:
http://80x24.org/spew/m/st-ll-v2-results%40whir.txt (good, I think)

Also http://80x24.org/spew/m/st-ccan-ll-fixup-1%4080x24.org.txt was
wrong and I rejected it due to improved tests in [Feature #10321]

#10 - 06/24/2015 08:20 AM - normalperson (Eric Wong)

- File 0001-st.c-use-ccan-linked-list-v2.patch added

Updated v2 patch.

I care about this more, now, since I want to try to make unordered hash an
option with st.c in the future for internals. This should make future changes
easier-to-understand with less code.

I'm willing to trade a little hash iteration (rare, I hope) performance
for better insert/delete performance (on big hashes).

Also, this has minor object size reductions (on 32-bit x86)

```
   text    data     bss     dec     hex filename
  14718      24       0   14742    3996 st.o-before
  14166      24       0   14190    376e st.o-after
```

#11 - 01/31/2018 08:12 AM - normalperson (Eric Wong)

- Status changed from Assigned to Closed

Files

0001-st.c-use-ccan-linked-list.patch	13.1 KB	09/22/2014	normalperson (Eric Wong)
0001-st.c-use-ccan-linked-list-v2.patch	16.8 KB	06/24/2015	normalperson (Eric Wong)