Ruby - Feature #5903

Optimize st_table (take 2)

01/17/2012 11:37 PM - funny_falcon (Yura Sokolov)

Status:	Closed	
Priority:	Normal	
Assignee:	nobu (Nobuyoshi Nakada)	
Target version:		

Description

Given some of preparations to this patches already merged into ruby-trunk,
I suggest patches for improving st_table second time (first were #5789):

1. Usage of packing for st_table of any kind, not only for numeric hashes.

Most of hashes, allocated during page render in Rails are smaller than 6 entries.
In fact, during rendering "Issues" page of Redmine, 40% of hashes not even grows
above 1 entry. They are small options hashes, passed to numerous helper methods.

This patch packs hashes upto 6 entries in a way like numeric hashes from trunk.
Also it pack hashes of size 0 and 1 into st_table inself, so that there is no
need to allocate any "bins" at all.

https://github.com/ruby/ruby/pull/84.patch
https://github.com/ruby/ruby/pull/84

1. Usage of specialized pool for allocating st_table, st_table_entry structures
   and st_table.bins of smallest size (11)

Usage of specialized pool for this allocations give great speedup for hash creation.
Also it gives countable reduction of memory consumption.

https://github.com/ruby/ruby/pull/83.patch
https://github.com/ruby/ruby/pull/83

First patch gives little overhead for creating hashes bigger than 6 entries when applied alone.
But both patches combined are not slower than ruby-trunk for hashes of any size.

Performance testing is here https://gist.github.com/1626602

Related issues:

Related to Ruby - Feature #12142: Hash tables with open addressing	**Closed**

History

#1 - 01/20/2012 09:48 PM - funny_falcon (Yura Sokolov)

I've updated pool_allocation patch to use more efficient pool algorithm.
No separate commit, I just rewrite branch.

#2 - 01/31/2012 02:18 AM - matz (Yukihiro Matsumoto)

I am positive about this patch. Nobu, could you review the patch and check it in unless you find any problem?

Matz.

#3 - 01/31/2012 07:07 AM - nobu (Nobuyoshi Nakada)

What's "_black_magick"?

#4 - 01/31/2012 09:23 AM - matz (Yukihiro Matsumoto)

Hi,

I meant 83.patch. I think 84.patch (pool allocation) requires more
discussion.

```
matz.
```

In message "Re: [ruby-core:42277] [ruby-trunk - Feature #5903] Optimize st_table (take 2)"
on Tue, 31 Jan 2012 07:07:15 +0900, Nobuyoshi Nakada nobu@ruby-lang.org writes:
|
|
|Issue #5903 has been updated by Nobuyoshi Nakada.
|
|
|What's "_black_magick"?

#5 - 01/31/2012 09:23 AM - matz (Yukihiro Matsumoto)

Hi,

In message "Re: [ruby-core:42279] Re: [ruby-trunk - Feature #5903] Optimize st_table (take 2)"
on Tue, 31 Jan 2012 08:59:06 +0900, Yukihiro Matsumoto matz@ruby-lang.org writes:

|I meant 83.patch. I think 84.patch (pool allocation) requires more
|discussion.

Oops, 83.patch was pool allocation. I meant 84.patch.

```
matz.
```

#6 - 01/31/2012 02:08 PM - funny_falcon (Yura Sokolov)

Nobuyoshi Nakada wrote:

> What's "_black_magick"?

On my computer, pool allocator work by 1% faster when I keep those two assignment to this "magic" field.

May be it is a case of my computer, cause when I remove heaps_freed from gc.c (objspace->heap.freed
http://bugs.ruby-lang.org/projects/ruby-trunk/repository/entry/gc.c#L412), it start to work slower too.
But it is not ever assigned (cause it is initialized with zero, and last < heaps_freed is never true
http://bugs.ruby-lang.org/projects/ruby-trunk/repository/entry/gc.c#L2170)

Yura

#7 - 01/31/2012 02:46 PM - funny_falcon (Yura Sokolov)

Hi,

I could work on questions about pool allocation.

Regards,

Yura.

#8 - 02/01/2012 01:05 PM - nobu (Nobuyoshi Nakada)

Another question about packing.
Why are PKEY_POS and PVAL_POS from the tail?

#9 - 02/01/2012 03:23 PM - funny_falcon (Yura Sokolov)

Nobuyoshi Nakada wrote:

> Another question about packing.
> Why are PKEY_POS and PVAL_POS from the tail?

It allows hash values to be very close to each other, so that while loop in find_packed_index runs through them very fast and does not touch another cache line of cpu.
And only when it found equal hash it jumps to check key. This allows searching in packed hash be even slightly faster than in not packed hash of same size.

Initially I experiment with variable sized packed hashes, so that num_bins is used and they goes from tail to avoid division by 3.
With fixed size this could be simplified.

I pushed a commit which places PKEY_POS and PVAL_POS after hashes, but in forward order.

They could be placed altogether (like i*3, i*3+1, i*3+2). remove_packed_entry should be changed accordantly. I think, this could improve iteration

over hash.

#10 - 02/05/2012 11:50 PM - funny_falcon (Yura Sokolov)

Table packing slows down st_foreach a bit, and GC suffers from it.
So that I add a st_foreach_nocheck to fix GC https://github.com/funny-falcon/ruby/commit/84d08af5d2943c3dd1a1d0c361fa22c2c7ae5ca4

#11 - 02/09/2012 11:47 PM - funny_falcon (Yura Sokolov)

I've updated table packing patch to correlate with trunk
https://github.com/ruby/ruby/pull/84
https://github.com/ruby/ruby/pull/84.patch

Also there is additional commit which increases usage of st_foreach_nocheck
https://github.com/funny-falcon/ruby/commit/95d6ddcb2b91
https://github.com/funny-falcon/ruby/commit/95d6ddcb2b91.diff

#12 - 03/25/2012 06:37 AM - funny_falcon (Yura Sokolov)

As far this ticket not closed, I'll post hash related patch here:

https://github.com/ruby/ruby/pull/107
https://github.com/ruby/ruby/pull/107.patch

1. remove some unused code from st.c and hash.c
2. change rb_hash_modify to rb_hash_modify_check when st_table allocation is not necessary
3. move part of safe iteration logic to st.c to make it clearer
   This is arguable change, cause it clearly do not have positive impact on performance,
   but make check consumes 592.2 second before this change and 595.4 after - less than 1 percent,
   so that I suppose, difference is negligible.
4. introduce st_shift to optimize Hash#shift

#13 - 03/27/2012 12:18 AM - funny_falcon (Yura Sokolov)

Add couple of commits to pull request https://github.com/ruby/ruby/pull/107 :

1. Removal of ST_CHECK .
   ST_CHECK were always returned instead of ST_CONTINUE inside of some st_foreach loops.
   Now such calls to st_foreach are converted to calls to st_foreach_check.
   So that, there is no reason to differentiate ST_CHECK and ST_CONTINUE, which simplifies calling code a bit.
   Also, it allows to simplify st_foreach_check code.
2. Traditionally, ultrapacking for empty and one element tables

#14 - 03/30/2012 12:43 AM - mame (Yusuke Endoh)

- Status changed from Open to Assigned

- Assignee set to nobu (Nobuyoshi Nakada)

#15 - 11/24/2012 01:09 PM - mame (Yusuke Endoh)

- Target version changed from 2.0.0 to 2.6

I'll postpone the pool allocation to next minor. Sorry.

--
Yusuke Endoh mame@tsg.ne.jp

#16 - 10/07/2016 01:35 PM - funny_falcon (Yura Sokolov)

I think, this could be closed in favor of https://bugs.ruby-lang.org/issues/12142

#17 - 10/31/2016 03:40 AM - shyouhei (Shyouhei Urabe)

- Related to Feature #12142: Hash tables with open addressing added

#18 - 10/31/2016 03:41 AM - shyouhei (Shyouhei Urabe)

- Status changed from Assigned to Closed

Closing. Follow-up issue is #12142.