Let round_capa for ID table not allocate excess capacity for power of 2 ints >= 4

right now round_capa value is rounded up to the next power of 2

```
round_capa(4) -> returns 8
round_capa(8) -> returns 16
round_capa(16) -> returns 32

round_capa(5) -> returns 8
round_capa(9) -> returns 16
round_capa(17) -> returns 32
etc.
```

it seems wasteful to allocate the extra items capacity, so this PR changes that to

```
round_capa(4) -> returns 4
round_capa(8) -> returns 8
round_capa(16) -> returns 16

round_capa(5) -> returns 8
round_capa(9) -> returns 16
round_capa(17) -> returns 32
etc.
```

the main purpose is to reduce memory usage especially during boot

my patch also uses BUILTIN_CLZ macro instead of shifts that makes it slightly faster

here's a benchmark

```
require 'benchmark/ips'

Benchmark.ips do |x|
  x.config(time: 20, warmup: 3)

  x.report('struct', "Struct.new(*('a'..'z').map { |x| x.to_sym })")
end

trunk
Warming up -----------------------------
  struct 527.000  i/100ms
Calculating -----------------------------
  struct  5.461k (± 5.5%) i/s - 109.089k in 20.040253s

methodmising - POW2_P (github)
Warming up -----------------------------
  struct 544.000  i/100ms
Calculating -----------------------------
  struct  5.570k (± 4.1%) i/s - 111.520k in 20.057245s

ahorek - BUILTIN_CLZ (id_table.c.patch)
Warming up -----------------------------
  struct 571.000  i/100ms
Calculating -----------------------------
  struct  5.812k (± 3.6%) i/s - 116.484k in 20.070607s
```
discussion [https://github.com/ruby/ruby/pull/2083](https://github.com/ruby/ruby/pull/2083)

**Associated revisions**

Revision 8e13da1e - 08/28/2019 02:29 AM - pavel
optimize get_power2 [Feature #15631]

Merged: [https://github.com/ruby/ruby/pull/2292](https://github.com/ruby/ruby/pull/2292)

**History**

### #1 - 03/05/2019 07:57 PM - ahorek (Pavel Rosický)
- Description updated

### #2 - 03/11/2019 10:49 AM - methodmissing (Lourens Naudé)
Thanks for raising this Pavel.

st_init_table_with_size(0) effectively also allocates additional capacity, but if and how quickly the hash tables mutate I'll investigate later.


A simple peek suggests a total table size of 152 bytes on init, but will investigate time to mutation of these 0 sized tables this evening:

```diff
diff --git a/st.c b/st.c
index ed235ce74e..f2b99d7771 100644
--- a/st.c
+++ b/st.c
@@ -615,6 +615,8 @@ st_init_table_with_size(const struct st_hash_type *type, st_index_t size)
     #ifdef ST_DEBUG
     st_check(tab);
     #endif
+    printf("# st_init_table_with_size(%d) \rightarrow %d (%d)\n", size, n, st_memsize(tab));
+    return tab;
 }

 linking miniruby
 # st_init_table_with_size(0) \rightarrow 2 (152)
```
funny_falcon, do you have any opinion?
if no opinion, we'll discuss it one month later and will commit it.
in fact, I can't check algorithm, so we can try it.

ahorek:
could you give me more performance measurements?
  • you should not call map in iteration (you should prepare IDs before)
  • now you only measures 26 fields. could you measure other numbers, 1 to 50, for example.

Thanks,
Koichi

Thanks for the review Koichi. I tested the patch on a rails app (redmine), but unfortunately there's no improvement.
0.1MB less memory after boot (150MB total)
No measurable difference in performance
I'll investigate the second case. Hash creation might be a better place to optimize.

I've attached the second patch for st_init_table_with_size. In theory it should be faster, but I can't measure any difference in ruby.

It should use SIZEOF_ST_INDEX_T and nlz(IntPtr).

Pavel added a new patch for get_power2 in https://github.com/ruby/ruby/pull/2292

I run several benchmark suites for both patches
https://github.com/ruby-bench/ruby-bench-suite
https://github.com/schneems/derailed_benchmarks

but all differences were within margin of error. Here's an optimized assembly comparsion that explains why:
saving 1-2 instructions makes no real difference.

feel free to close the issue. Thanks

### #11 - 08/28/2019 03:08 AM - Anonymous

- Status changed from Open to Closed

Applied in changeset g988e13da1ee83028000e5d719f9526379a32765a81.

---

optimize get_power2 [Feature #15631]

Merged: https://github.com/ruby/ruby/pull/2292

**Files**

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>id_table.c.patch</td>
<td>534 Bytes</td>
<td>03/01/2019</td>
<td>ahor (Pavel Rosický)</td>
</tr>
<tr>
<td>st.c.patch</td>
<td>455 Bytes</td>
<td>07/19/2019</td>
<td>ahor (Pavel Rosický)</td>
</tr>
<tr>
<td>st.c.patch</td>
<td>434 Bytes</td>
<td>07/21/2019</td>
<td>ahor (Pavel Rosický)</td>
</tr>
</tbody>
</table>