Ruby master - Feature #641

GC patch to cache the most recent heap for is_pointer_to_heap

10/12/2008 03:40 PM - rogerdpack (Roger Pack)

Status: Rejected
Priority: Normal
Assignee: authorNari (Narihiro Nakamura)
Target version: 2.0.0

Description

This patch seems benign and speeds up gc a bit. In a completely contrived test [meant to hammer the GC]:

```ruby
require 'benchmark'
require 'pp'
a = []
1_000_000.times { a << [3] }
pp Benchmark.measure { 33.times { GC.start } }

speed went from
@real=1.99618101119995,
to
@real=1.75,

[I think Hongli does something similar with bit field lookups]. A similar patch might work for 1.8.6
Thanks!
```

History

#1 - 10/14/2008 06:53 AM - rogerdpack (Roger Pack)

```
=begin
This appears to actually slow down 1.8.6 [but speedup 1.9--I'd guess because 1.9 has those fixed heap sizes]. in 1.8.6 TRUNK it takes 1.6s [faster, for some reason].
=>R
==end
```

#2 - 10/25/2008 11:10 AM - authorNari (Narihiro Nakamura)

```
=begin
Hi.

It seems interesting :)

I tried it with 1.9
But this not appeared speed up.
```

```bash
$ cat bm_ruby-core_19301.rb
require 'benchmark'
require 'pp'
GC::Profiler.enable
a = []
1_000_000.times { a << [3] }
pp Benchmark.measure { 33.times { GC.start } }
GC::Profiler.report
```

```bash
$ ./ruby-trunk/ruby bm_ruby-core_19301.rb
#
Index  Invoke Time(sec)  Use Size(byte)  Total Size(byte)  Total Object  GC Time(ms)
1     0.032           539600         622592            31122      0.00000000000000028257
2     0.040         1080240        1097728            54873      0.40000000000000018874
3     0.052         1931380        1949696           97461      0.39999999999999968914
4     0.076         3469980        3489792          174447      1.20009999999999950049
```

08/06/2021
<table>
<thead>
<tr>
<th>Index</th>
<th>Invoke Time(sec)</th>
<th>Use Size(byte)</th>
<th>Total Size(byte)</th>
<th>Total Object</th>
<th>GC Time(ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.032</td>
<td>539940</td>
<td>622592</td>
<td>31122</td>
<td>0.39999999999999991118</td>
</tr>
<tr>
<td>2</td>
<td>0.044</td>
<td>1080320</td>
<td>1097728</td>
<td>54873</td>
<td>0.40000000000000007772</td>
</tr>
<tr>
<td>3</td>
<td>0.056</td>
<td>1931440</td>
<td>1949696</td>
<td>97461</td>
<td>0.40010000000000003446</td>
</tr>
<tr>
<td>4</td>
<td>0.084</td>
<td>3470040</td>
<td>3489792</td>
<td>174447</td>
<td>1.200099999999995049</td>
</tr>
<tr>
<td>5</td>
<td>0.132</td>
<td>6236240</td>
<td>6258688</td>
<td>312858</td>
<td>2.00010000000000021103</td>
</tr>
<tr>
<td>6</td>
<td>0.220</td>
<td>11212100</td>
<td>11239424</td>
<td>561834</td>
<td>3.60020000000000006679</td>
</tr>
<tr>
<td>7</td>
<td>0.376</td>
<td>20181760</td>
<td>20217856</td>
<td>101646</td>
<td>6.40039999999999746868</td>
</tr>
<tr>
<td>8</td>
<td>0.440</td>
<td>20225800</td>
<td>20335244</td>
<td>1016379</td>
<td>6.40039999999999746868</td>
</tr>
<tr>
<td>9</td>
<td>0.590</td>
<td>20489204</td>
<td>20519924</td>
<td>1017198</td>
<td>6.80049999999999680727</td>
</tr>
<tr>
<td>10</td>
<td>0.572</td>
<td>20225580</td>
<td>20365312</td>
<td>1018017</td>
<td>6.80039999999999800195</td>
</tr>
<tr>
<td>11</td>
<td>0.640</td>
<td>20225580</td>
<td>20381696</td>
<td>1018836</td>
<td>7.200400000000000279812</td>
</tr>
<tr>
<td>12</td>
<td>0.712</td>
<td>20225580</td>
<td>20398080</td>
<td>1019655</td>
<td>6.800499999999999510436</td>
</tr>
<tr>
<td>13</td>
<td>0.780</td>
<td>20225580</td>
<td>20414464</td>
<td>1020474</td>
<td>6.800399999999999711378</td>
</tr>
<tr>
<td>14</td>
<td>0.848</td>
<td>20225580</td>
<td>20430848</td>
<td>1021293</td>
<td>6.400000000000000031122</td>
</tr>
<tr>
<td>15</td>
<td>0.912</td>
<td>20225580</td>
<td>20447232</td>
<td>1022112</td>
<td>6.4000000000000000498215</td>
</tr>
<tr>
<td>16</td>
<td>0.976</td>
<td>20225580</td>
<td>20463616</td>
<td>1022931</td>
<td>6.80049999999999933102</td>
</tr>
<tr>
<td>17</td>
<td>1.044</td>
<td>20225580</td>
<td>20480000</td>
<td>1023750</td>
<td>6.800399999999999711378</td>
</tr>
<tr>
<td>18</td>
<td>1.112</td>
<td>20225580</td>
<td>20496384</td>
<td>1024569</td>
<td>7.200500000000000167688</td>
</tr>
<tr>
<td>19</td>
<td>1.184</td>
<td>20225580</td>
<td>20512768</td>
<td>1025388</td>
<td>6.800399999999999089653</td>
</tr>
<tr>
<td>20</td>
<td>1.252</td>
<td>20225580</td>
<td>20529152</td>
<td>1026207</td>
<td>6.800399999999999533742</td>
</tr>
<tr>
<td>21</td>
<td>1.320</td>
<td>20225580</td>
<td>20545536</td>
<td>1027026</td>
<td>7.2004999999999995825762</td>
</tr>
<tr>
<td>22</td>
<td>1.392</td>
<td>20225580</td>
<td>20561920</td>
<td>1027845</td>
<td>6.8004000000000000217659</td>
</tr>
<tr>
<td>23</td>
<td>1.468</td>
<td>20225580</td>
<td>20584484</td>
<td>1028449</td>
<td>6.800399999999999711378</td>
</tr>
<tr>
<td>24</td>
<td>2.080</td>
<td>20225580</td>
<td>20620224</td>
<td>1029267</td>
<td>6.8004000000000000658277</td>
</tr>
<tr>
<td>25</td>
<td>2.152</td>
<td>20225580</td>
<td>20642144</td>
<td>1030854</td>
<td>6.4000000000000000217659</td>
</tr>
<tr>
<td>26</td>
<td>2.216</td>
<td>20225580</td>
<td>20675828</td>
<td>1031673</td>
<td>6.8005000000000000398614</td>
</tr>
</tbody>
</table>

$ ./ruby1.9_cach/ruby bm_ruby-core_19301.rb

# GC 42 invokes.
Trying it on Linux I get about the difference between 1.84s [current gc.c] and 1.83s [patched].

I think that we can see more difference if we use more memory. Changing the test file's line
1_000_000.times { a << [3] }
to be
10_000_000.times { a << [3] }
on Linux for me this shows
new:
@real=17.7093026638031,
normal:
@real=18.3347587585449

so...overall I'd call it a small improvement.

Some things to note: you may have to run the test files several times to get a "best" result, and also [note to myself] I always have to run make install
after making changes because for some reason it relies on previously installed libraries, despite the local code being different.

Thanks!
--Roger

#4 - 11/29/2008 04:21 PM - ko1 (Koichi Sasada)
- Assignee set to matz (Yukihiro Matsumoto)

=begin

#5 - 12/11/2008 12:20 PM - yugui (Yuki Sonoda)
- Target version set to 2.0.0

=begin

#6 - 09/14/2010 04:53 PM - shyouhei (Shyouhei Urabe)
- Status changed from Open to Assigned

=begin

#7 - 02/07/2012 09:05 PM - mame (Yusuke Endoh)
- Status changed from Assigned to Rejected
- Assignee changed from matz (Yukihiro Matsumoto) to authorNari (Narihiro Nakamura)

Hello,

I'm rejecting this feature ticket because no progress has been made for a long time. See [ruby-core:42391].

Nari, are you willing to treat this ticket? If you are, please reopen this. Maybe because lazy sweeping was imported, the patch does not work now and it seems not to be trivial to fix it.

--

#3 - 10/28/2008 03:18 PM - rogerdpack (Roger Pack)
=begin

Trying it on Linux I get about the difference between 1.84s [current gc.c] and 1.83s [patched].

I think that we can see more difference if we use more memory. Changing the test file's line
1_000_000.times { a << [3] }
to be
10_000_000.times { a << [3] }
on Linux for me this shows
new:
@real=17.7093026638031,
normal:
@real=18.3347587585449

so...overall I'd call it a small improvement.

Some things to note: you may have to run the test files several times to get a "best" result, and also [note to myself] I always have to run make install
after making changes because for some reason it relies on previously installed libraries, despite the local code being different.

Thanks!
--Roger

=begin
<table>
<thead>
<tr>
<th>Files</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache_latest_heap.diff</td>
<td>1.27 KB</td>
<td>10/12/2008</td>
<td>rogerdpack (Roger Pack)</td>
</tr>
</tbody>
</table>