Ruby master - Feature #9638

[PATCH] limit IDs to 32-bits on 64-bit systems

03/14/2014 07:07 PM - normalperson (Eric Wong)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Rejected</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td></td>
</tr>
<tr>
<td>Target version:</td>
<td>2.2.0</td>
</tr>
</tbody>
</table>

Description

This should allow better use of cache-friendly lookup mechanisms such as funny_falcon's sparse array in [ruby-core:55079]

Also limits symbol space to prevent OOM.

Some structs may also be made smaller as a result (rb_method_entry_t).

We're changing ABI for 2.2.0 anyways, so this is a good time to introduce this change.

Related issues:

Related to Ruby master - Feature #11420: Introduce ID key table into MRI
Closed

History

#1 - 03/14/2014 07:08 PM - normalperson (Eric Wong)

sparse array is described in ruby-core:55079

#2 - 05/26/2014 05:38 AM - normalperson (Eric Wong)

I'm not sure if this is possible anymore due to SymbolGC
No big deal, though.

#3 - 05/27/2014 02:45 AM - ngoto (Naohisa Goto)

I'm using machines that have 2TB or more main memory. I think the machines can treat more than $2^{32}$ symbols and I want to use full 64-bit capacity.

#4 - 05/27/2014 03:22 AM - normalperson (Eric Wong)

I am OK with closing this issue (but I'm not sure if I have permissions to close on redmine).

However, your applications need more than $2^{32}$ different symbols?
That scares me :

How much memory do your Ruby processes use?

The Symbol table currently takes at least (48 + 48 + 40 = 136) bytes per symbol on 64-bit, so $136 \times 2^{32}$ is 544 gigabytes just for the symbol table (w/fstrings) in your app. That does not even account for memory of symbols with string representations longer than 23 bytes, nor the memory for hash table buckets.

I need to know because I am also looking into using khash[1] for the symbol table. By default, khash internal buckets/counters are all 32-bits. We can tweak khash to use 64-bit counters if needed, but $2^{32}$ symbols really should be enough.

The symbol table with khash might reduce memory overhead to ~90 bytes per-symbol on average, though...

mrbuy also uses khash for (all?) its hash table needs.

#5 - 05/28/2014 01:02 AM - normalperson (Eric Wong)

- Status changed from Open to Rejected
Also limits symbol space to prevent OOM.

What is OOM?
Out of memory?

Symbol GC doesn't help?

--
// SASADA Koichi at atdot dot net

Yes, out-of-memory.
Symbol GC doesn't help?

It does; but OOM was a secondary concern of mine.

I mainly wanted 32-bit ID so it might be easier to pack some structs on 64-bit machines. 64-bit ID is not a big issue, though.

- Related to Feature #11420: Introduce ID key table into MRI added

Files

0001-ID-is-always-uint32_t.patch 3.62 KB 03/14/2014 normalperson (Eric Wong)