Ruby master - Misc #10628
Performance of URI module
12/21/2014 02:43 PM - tgxworld (Guo Xiang Tan)

<table>
<thead>
<tr>
<th>Status:</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>naruse (Yui NARUSE)</td>
</tr>
</tbody>
</table>

**Description**

Please view attached screenshot or go to [the following link](#) to see benchmark graph over time.

It got slower after this [commit](#).

Hope this helps.

**Associated revisions**

**Revision aa93c62e - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@48980 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

**Revision 48980 - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


**Revision 48980 - 12/24/2014 11:50 PM - normalperson (Eric Wong)**

lib/uri: performance improvements [misc #10628]


**Revision 48980 - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


**Revision 48980 - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


**Revision 48980 - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


**Revision 48980 - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


**Revision 48980 - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


**Revision 48980 - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


**Revision 48980 - 12/24/2014 11:50 PM - normal**

lib/uri: performance improvements [misc #10628]


09/04/2021
Revision 48980 - 12/24/2014 11:50 PM - normal

lib/uri: performance improvements [misc #10628]


History

#1 - 12/22/2014 10:08 PM - normalperson (Eric Wong)

tgx_world: Please don't use screenshots when a few words explaining the situation will do. Not everyone can view pics: lack of working GUI, poor eyesight or danger of opening inappropriate images in public areas.

Fortunately, I was able to open the link to see the issue was with the bm_app_uri benchmark being slower.

I tried some optimizations, but it's still slower than 2.1.5 on my Haswell E3-1230 v3 server:

http://80x24.org/spew/m/ugly-uri-optimizations@r48922.txt

Unfortunately, these make evaluating effects of any future compile.c optimizations more difficult.

benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name 2.1.5 trunk built
app_uri 0.484 0.584 0.500

Speedup ratio: compare with the result of `2.1.5' (greater is better)
name trunk built
app_uri 0.829 0.968

naruse: any thoughts?

#2 - 12/23/2014 12:47 AM - tgxworld (Guo Xiang Tan)

Eric Wong wrote:

tgx_world: Please don't use screenshots when a few words explaining the situation will do. Not everyone can view pics: lack of working GUI, poor eyesight or danger of opening inappropriate images in public areas.

Fortunately, I was able to open the link to see the issue was with the bm_app_uri benchmark being slower.

I tried some optimizations, but it's still slower than 2.1.5 on my Haswell E3-1230 v3 server:

http://80x24.org/spew/m/ugly-uri-optimizations@r48922.txt

Unfortunately, these make evaluating effects of any future compile.c optimizations more difficult.

benchmark results:
minimum results in each 10 measurements.
Execution time (sec)
name 2.1.5 trunk built
app_uri 0.484 0.584 0.500

Speedup ratio: compare with the result of `2.1.5' (greater is better)
name trunk built
app_uri 0.829 0.968

naruse: any thoughts?

Opps got it. I'll try to explain better in words next time.
The benchmark consists about two parts: split and new.

In split, the difference of its regexp seems the cause of slow down (for example backtracks of userinfo. It can be optimized.

In new, some checks are always enabled; it is feature.

```ruby
require 'uri'
100_000.times{
  scheme, userinfo, host, port,
  registry, path, opaque, query, fragment = URI.split('http://www.ruby-lang.org')
  URI::HTTP.new(scheme, userinfo, host, port,
  registry, path, opaque, query,
  fragment, URI,false)
}
```

Anyway I'm ok for Eric's patch.
Could you commit it?

We should consider ext/uri or strict/loose API for Ruby 2.3.

We should be able to improve compile.c to do more optimizations.
I prefer to avoid any new C-exts.

Things like Feature #10423.
Also, this patch gave me a new ideas: we should be able to merge/share nearby call_info structs on the same object. For example:

```ruby
m["a"], m["b"], m["c"]
```

May all use the same call_info object, I think:

```ruby
{:mid=>[:], :flag=>256, :blockptr=>nil, :orig_argc=>1},
```

This can improve IC hit rates and reduce memory.

Latest results so far after the patch :)  
https://railsbench.herokuapp.com/tgxworld/ruby?utf8=%E2%9C%93&result_types%5B%5D=app_uri&commit=Submit

Do let me know if you need any more information. Happy holidays!

I actually can't see results in my text-only browser, just that bm_app_uri is selected. Can you just show us numbers? Thanks.
Eric Wong wrote:

tgx_world@hotmail.com wrote:

Latest results so far after the patch :)  

https://railsbench.herokuapp.com/tgxworld/ruby?utf8=%E2%9C%93&result_types%5B%5D=app_uri&commit=Submit  

Do let me know if you need any more information. Happy holidays!

I actually can’t see results in my text-only browser, just that bm_app_uri is selected. Can you just show us numbers? Thanks.

On my builder, these are the following results:  
before regression: 0.97  
regression: 1.1 ~ 1.2  
after patch: 1.0

<table>
<thead>
<tr>
<th>Files</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Screenshot from 2014-12-21 22_41_39.png</td>
<td>63.5 KB</td>
<td>12/21/2014</td>
<td>tgxworld (Guo Xiang Tan)</td>
</tr>
</tbody>
</table>