Ruby master - Feature #13368

Improve performance of Array#sum with float elements

03/27/2017 03:50 AM - watson1978 (Shizuo Fujita)

Status: Closed
Priority: Normal
Assignee: mrkn (Kenta Murata)
Target version:

Description

The declaration of local variable in loop, it will initialize local variable for each run of the loop with clang generated code. So, it shouldn't declare the local variable in heavy loop.

Array#sum with float elements will be faster around 30%.

Before

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.320000</td>
<td>0.010000</td>
<td>3.330000</td>
<td>( 3.336088)</td>
</tr>
</tbody>
</table>

After

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.590000</td>
<td>0.010000</td>
<td>2.600000</td>
<td>( 2.602399)</td>
</tr>
</tbody>
</table>

Test code

```ruby
require 'benchmark'

Benchmark.bmbm do |x|
  ary = []
  10000.times { ary << Random.rand }

  x.report do
    50000.times do
      ary.sum
    end
  end
end
```

Patch

https://github.com/ruby/ruby/pull/1555

History

#1 - 04/10/2017 07:03 PM - normalperson (Eric Wong)
+Cc ruby-core since the original post was English

watson1978@gmail.com wrote:

Issue #13368 has been reported by watson1978 (Shizuo Fujita).

Bug #13368: Improve performance of Array#sum with float elements
https://bugs.ruby-lang.org/issues/13368

The declaration of local variable in loop, it will initialize local variable for each run of the loop with clang generated code. So, it shouldn't declare the local variable in heavy loop.

I disagree with this change. Different compilers optimize differently.
Disclaimer: I used commit 3635f883096604d0e6453dc9d2484d5c92467109 after having:

```bash
fetch = +refs/pull/*:refs/remotes/ruby/pull/*
```

In the `[remote ]` section of my `.git/config` and did not use any proprietary JavaScript or browser to apply your change.

#2 - 04/10/2017 07:03 PM - normalperson (Eric Wong)

+Cc ruby-core since the original post was English

watson1978@gmail.com wrote:

Issue #13368 has been reported by watson1978 (Shizuo Fujita).

---

Bug #13368: Improve performance of Array#sum with float elements
https://bugs.ruby-lang.org/issues/13368

The declaration of local variable in loop, it will initialize local variable for each run of the loop with clang generated code. So, it shouldn't declare the local variable in heavy loop.

I disagree with this change. Different compilers optimize differently.

Patch

https://github.com/ruby/ruby/pull/1555

Disclaimer: I used commit 3635f883096604d0e6453dc9d2484d5c92467109 after having:

```bash
fetch = +refs/pull/*:refs/remotes/ruby/pull/*
```

In the `[remote ]` section of my `.git/config` and did not use any proprietary JavaScript or browser to apply your change.

#3 - 04/14/2017 06:45 AM - watson1978 (Shizuo Fujita)

When I filed this ticket, I tried to run benchmark on macOS + clang only. Then, I tried to do on 2 environments in additional.

I found what my patch is effective with clang environment only (such as macOS or FreeBSD which use clang as default compiler).

**macOS 10.12 + gcc 6.3.0**

**Before**

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.750000</td>
<td>0.000000</td>
<td>2.750000</td>
<td>(2.757864)</td>
</tr>
</tbody>
</table>

**After**

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.730000</td>
<td>0.010000</td>
<td>2.740000</td>
<td>(2.740204)</td>
</tr>
</tbody>
</table>

**Ubuntu 16.04.4 + gcc 5.4.0**

**Before**

<table>
<thead>
<tr>
<th>user</th>
<th>system</th>
<th>total</th>
<th>real</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.400000</td>
<td>0.000000</td>
<td>2.400000</td>
<td>(2.395856)</td>
</tr>
</tbody>
</table>

**After**
watson1978 (Shizuo Fujita) wrote:

When I filed this ticket, I tried to run benchmark on macOS + clang only. Then, I tried to do on 2 environments in additional.

I found what my patch is effective with clang environment only (such as macOS or FreeBSD which use clang as default compiler).

Hmm, the result shows clang is not efficient as gcc yet this time. And improvement should be done by clang, not us. Or should be done by something cleaner patch.

naruse (Yui NARUSE) wrote:

Hmm, the result shows clang is not efficient as gcc yet this time. And improvement should be done by clang, not us. Or should be done by something cleaner patch.

Could you report this to clang developer if you don't want to fix this?

watson1978 (Shizuo Fujita) wrote:

Hmmm, the result shows clang is not efficient as gcc yet this time. And improvement should be done by clang, not us. Or should be done by something cleaner patch.

Could you report this to clang developer if you don't want to fix this?

Though I reported to clang when it breaks build like https://bugs.llvm.org/show_bug.cgi?id=8319
Clang is considered still development at performance like other inline threshold ones. They should improve more general cases.

jeremyevans0 (Jeremy Evans)
- Backport deleted (2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN)
- Tracker changed from Bug to Feature

hsbt (Hiroshi SHIBATA)
- Assignee set to mrkn (Kenta Murata)
- Status changed from Open to Closed

https://github.com/ruby/ruby/pull/1555 was merged at https://github.com/ruby/ruby/commit/2d001003e4b3a6c20ead09ed54b6726a7669f457