Ruby master - Bug #13085

io.c io_fwrite creates garbage

12/29/2016 01:11 AM - normalperson (Eric Wong)

Status:	Closed		
Priority:	Normal		
Assignee:	normalperson (Eric Wong)		
Target version:			
ruby -v:		**Backport:**	2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

Relying on rb_str_new_frozen for unconverted strings does not save memory because copy-on-write is always triggered in read-write I/O loops were subsequent IO#read calls will clobber the given write buffer.

```
buf = ''.b
while input.read(16384, buf)
  output.write(buf)
end
```

This generates a lot of garbage starting with Ruby 2.2 (r44471). For my use case, even IO.copy_stream generates garbage, since I wrap "write" to do Digest calculation in a single pass.

I tried using rb_str_replace and reusing the string as a hidden (klass == 0) thread-local, but rb_str_replace attempts CoW optimization by creating new frozen objects, too:

https://80x24.org/spew/20161229004417.12304-1-e@80x24.org/raw

So, I'm not sure what to do, temporal locking seems wrong for writing strings (I guess it's for reading?). I get test_threaded_flush failures with the following:

https://80x24.org/spew/20161229005701.9712-1-e@80x24.org/raw

IO#syswrite has the same problem with garbage. I can use IO#write_nonblock on fast filesystems while holding GVL, I guess...

Related issues:

Related to Ruby master - Bug #13299: backport r57469, r57472, r57508 (garbage...	**Closed**

Associated revisions

Revision 5c988df0 - 01/30/2017 09:54 PM - normal

string.c (rb_str_tmp_frozen_release): release embedded strings

Handle the embedded case first, since we may have an embedded duplicate and non-embedded original string.

- string.c (rb_str_tmp_frozen_release): handled embedded strings
- test/ruby/test_io.rb (test_write_no_garbage): new test [ruby-core:78898] [Bug #13085]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57471 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 57471 - 01/30/2017 09:54 PM - normalperson (Eric Wong)

string.c (rb_str_tmp_frozen_release): release embedded strings

Handle the embedded case first, since we may have an embedded duplicate and non-embedded original string.

- string.c (rb_str_tmp_frozen_release): handled embedded strings
- test/ruby/test_io.rb (test_write_no_garbage): new test [ruby-core:78898] [Bug #13085]

Revision 57471 - 01/30/2017 09:54 PM - normal

string.c (rb_str_tmp_frozen_release): release embedded strings

Handle the embedded case first, since we may have an embedded
duplicate and non-embedded original string.

- string.c (rb_str_tmp_frozen_release): handled embedded strings
- test/ruby/test_io.rb (test_write_no_garbage): new test [ruby-core:78898] [Bug #13085]

Revision 57471 - 01/30/2017 09:54 PM - normal

string.c (rb_str_tmp_frozen_release): release embedded strings

Handle the embedded case first, since we may have an embedded
duplicate and non-embedded original string.

- string.c (rb_str_tmp_frozen_release): handled embedded strings
- test/ruby/test_io.rb (test_write_no_garbage): new test [ruby-core:78898] [Bug #13085]

Revision 4b9a21cd - 01/30/2017 10:03 PM - normal

io.c (rb_io_syswrite): avoid leaving garbage after write

As with IO#write, IO#syswrite also generates garbage which can
be harmful in hand-coded read-write loops.

- io.c (swrite_arg, swrite_do, swrite_end): new (rb_io_syswrite): use new functions to cleanup garbage [ruby-core:78898] [Bug #13085]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@57472 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 57472 - 01/30/2017 10:03 PM - normalperson (Eric Wong)

io.c (rb_io_syswrite): avoid leaving garbage after write

As with IO#write, IO#syswrite also generates garbage which can
be harmful in hand-coded read-write loops.

- io.c (swrite_arg, swrite_do, swrite_end): new (rb_io_syswrite): use new functions to cleanup garbage [ruby-core:78898] [Bug #13085]

Revision 57472 - 01/30/2017 10:03 PM - normal

io.c (rb_io_syswrite): avoid leaving garbage after write

As with IO#write, IO#syswrite also generates garbage which can
be harmful in hand-coded read-write loops.

- io.c (swrite_arg, swrite_do, swrite_end): new (rb_io_syswrite): use new functions to cleanup garbage [ruby-core:78898] [Bug #13085]

Revision 57472 - 01/30/2017 10:03 PM - normal

io.c (rb_io_syswrite): avoid leaving garbage after write

As with IO#write, IO#syswrite also generates garbage which can
be harmful in hand-coded read-write loops.

- io.c (swrite_arg, swrite_do, swrite_end): new (rb_io_syswrite): use new functions to cleanup garbage [ruby-core:78898] [Bug #13085]

Revision baf330be - 03/12/2017 06:15 PM - naruse (Yui NARUSE)

merge revision(s) 57469,57471,57472,57503,57508: [Backport #13299]

```
    io.c: recycle garbage on write

    * string.c (STR_IS_SHARED_M): new flag to mark shared mulitple times
      (STR_SET_SHARED): set STR_IS_SHARED_M
      (rb_str_tmp_frozen_acquire, rb_str_tmp_frozen_release): new functions
      (str_new_frozen): set/unset STR_IS_SHARED_M as appropriate
    * internal.h: declare new functions
    * io.c (fwrite_arg, fwrite_do, fwrite_end): new
      (io_fwrite): use new functions
```

```
    Introduce rb_str_tmp_frozen_acquire and rb_str_tmp_frozen_release
    to manage a hidden, frozen string.  Reuse one bit of the embed
    length for shared strings as STR_IS_SHARED_M to indicate a string
    has been shared multiple times.  In the common case, the string
    is only shared once so the object slot can be reclaimed immediately.

    minimum results in each 3 measurements. (time and size)

    Execution time (sec)
    name                            trunk   built
    io_copy_stream_write            0.682   0.254
    io_copy_stream_write_socket     1.225   0.751

    Speedup ratio: compare with the result of `trunk' (greater is better)
    name    built
    io_copy_stream_write            2.680
    io_copy_stream_write_socket     1.630

    Memory usage (last size) (B)
    name                            trunk           built
    io_copy_stream_write            95436800.000    6512640.000
    io_copy_stream_write_socket     117628928.000   7127040.000

    Memory consuming ratio (size) with the result of `trunk' (greater is better)
    name    built
    io_copy_stream_write            14.654
    io_copy_stream_write_socket     16.505
    string.c (rb_str_tmp_frozen_release): release embedded strings

    Handle the embedded case first, since we may have an embedded
    duplicate and non-embedded original string.

    * string.c (rb_str_tmp_frozen_release): handled embedded strings
    * test/ruby/test_io.rb (test_write_no_garbage): new test
      [ruby-core:78898] [Bug #13085]
    io.c (rb_io_syswrite): avoid leaving garbage after write

    As with IO#write, IO#syswrite also generates garbage which can
    be harmful in hand-coded read-write loops.

    * io.c (swrite_arg, swrite_do, swrite_end): new
      (rb_io_syswrite): use new functions to cleanup garbage
      [ruby-core:78898] [Bug #13085]
    Add class name to assert messages
    io.c: remove rb_ensure usage for rb_str_tmp_frozen_* calls

    Using rb_ensure pessimizes the common case and makes the code
    more difficult to read and follow.  If we hit an exceptions
    during write, just let the GC handle cleanup as the exception
    is already bad for garbage.

    * io.c (io_fwrite): call rb_str_tmp_frozen{acquire,release} directly
      (rb_io_syswrite): ditto
      (fwrite_do, fwrite_end, swrite_do, swrite_end): remove
```

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/branches/ruby_2_4@57941 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 57941 - 03/12/2017 06:15 PM - naruse (Yui NARUSE)

merge revision(s) 57469,57471,57472,57503,57508: [Backport #13299]

```
io.c: recycle garbage on write

* string.c (STR_IS_SHARED_M): new flag to mark shared mulitple times
  (STR_SET_SHARED): set STR_IS_SHARED_M
  (rb_str_tmp_frozen_acquire, rb_str_tmp_frozen_release): new functions
  (str_new_frozen): set/unset STR_IS_SHARED_M as appropriate
* internal.h: declare new functions
* io.c (fwrite_arg, fwrite_do, fwrite_end): new
  (io_fwrite): use new functions

Introduce rb_str_tmp_frozen_acquire and rb_str_tmp_frozen_release
to manage a hidden, frozen string.  Reuse one bit of the embed
length for shared strings as STR_IS_SHARED_M to indicate a string
has been shared multiple times.  In the common case, the string
```

```
is only shared once so the object slot can be reclaimed immediately.

minimum results in each 3 measurements. (time and size)

Execution time (sec)
name                            trunk   built
io_copy_stream_write            0.682   0.254
io_copy_stream_write_socket     1.225   0.751

Speedup ratio: compare with the result of `trunk' (greater is better)
name    built
io_copy_stream_write            2.680
io_copy_stream_write_socket     1.630

Memory usage (last size) (B)
name                            trunk           built
io_copy_stream_write            95436800.000    6512640.000
io_copy_stream_write_socket     117628928.000   7127040.000

Memory consuming ratio (size) with the result of `trunk' (greater is better)
name    built
io_copy_stream_write            14.654
io_copy_stream_write_socket     16.505
string.c (rb_str_tmp_frozen_release): release embedded strings

Handle the embedded case first, since we may have an embedded
duplicate and non-embedded original string.

* string.c (rb_str_tmp_frozen_release): handled embedded strings
* test/ruby/test_io.rb (test_write_no_garbage): new test
  [ruby-core:78898] [Bug #13085]
io.c (rb_io_syswrite): avoid leaving garbage after write

As with IO#write, IO#syswrite also generates garbage which can
be harmful in hand-coded read-write loops.

* io.c (swrite_arg, swrite_do, swrite_end): new
  (rb_io_syswrite): use new functions to cleanup garbage
  [ruby-core:78898] [Bug #13085]
Add class name to assert messages
io.c: remove rb_ensure usage for rb_str_tmp_frozen_* calls

Using rb_ensure pessimizes the common case and makes the code
more difficult to read and follow.  If we hit an exceptions
during write, just let the GC handle cleanup as the exception
is already bad for garbage.

* io.c (io_fwrite): call rb_str_tmp_frozen{acquire,release} directly
  (rb_io_syswrite): ditto
  (fwrite_do, fwrite_end, swrite_do, swrite_end): remove
```

History

#1 - 12/31/2016 01:01 AM - normalperson (Eric Wong)

- File 0001-io.c-io_fwrite-temporarily-freeze-string-when-writin.patch added

Proposed patch to temporarily freeze string while copying

```
    io.c (io_fwrite): temporarily freeze string when writing

    This avoids garbage from IO#write for [Bug #13085].
    Memory usage from benchmark/bm_io_copy_stream_write.rb
    is reduced greatly:

      target 0: a (ruby 2.5.0dev (2016-12-30 trunk 57236) [x86_64-linux])
      target 1: b (ruby 2.5.0dev (2016-12-30) [x86_64-linux])

      Memory usage (last size) (B)
      name  a       b
      io_copy_stream_write  82235392.000    6651904.000

      Memory consuming ratio (size) with the result of `a' (greater is better)
      name  b
      io_copy_stream_write  12.363
```

```
    There is also a speedup in execution time:

      Execution time (sec)
      name  a       b
      io_copy_stream_write  0.380   0.143

      Speedup ratio: compare with the result of `a' (greater is better)
      name  b
      io_copy_stream_write  2.651

    Caveat, there is one potential race condition:

    If another thread calls String#freeze on the string we are
    currently writing; we will blindly unfreeze it during
    fwrite_unfreeze from ensure.  However, I do not expect this to
    be a real-world case.

    Ideally, Ruby should have a way of detecting threads which
    are not visible to other threads.
```

#2 - 01/05/2017 01:02 AM - normalperson (Eric Wong)

normalperson@yhbt.net wrote:

> Issue #13085 has been updated by Eric Wong.
>
> File 0001-io.c-io_fwrite-temporarily-freeze-string-when-writin.patch added
>
> ```
> Caveat, there is one potential race condition:
>
> If another thread calls String#freeze on the string we are
> currently writing; we will blindly unfreeze it during
> fwrite_unfreeze from ensure.  However, I do not expect this to
> be a real-world case.
>
> Ideally, Ruby should have a way of detecting threads which
> are not visible to other threads.
> ```

Any comment? I still have not been able to think of a better
solution. I'll commit in a month or five if no objections.

> https://bugs.ruby-lang.org/issues/13085#change-62335

#3 - 01/05/2017 01:16 AM - nobu (Nobuyoshi Nakada)

No.
Your patch releases the GVL while unfreezing the string to be written.
If the write operation is blocked, other threads can clobber that string.

#4 - 01/05/2017 03:31 AM - normalperson (Eric Wong)

nobu@ruby-lang.org wrote:

> No.
> Your patch releases the GVL while unfreezing the string to be written.
> If the write operation is blocked, other threads can clobber that string.

It does?

fwrite_unfreeze happens in rb_ensure e_proc, which has GVL.

From what I can tell, GVL release happens only inside
fwrite_freeze (rb_ensure b_proc):

```
    fwrite_freeze (rb_ensure b_proc)
        io_binwrite
            rb_mutex_synchronize(fptr->write_lock...)
                rb_mutex_lock = NOGVL
                io_binwrite_string (identical as below:)
            io_binwrite_string
```

```
                rb_writev_internal
                    rb_thread_io_blocking_region = NOGVL
                rb_write_internal
                    rb_thread_io_blocking_region = NOGVL
            rb_io_wait_writable = NOGVL
```

By the time e_proc (fwrite_unfreeze) is called by rb_ensure
we have GVL, again.

#5 - 01/05/2017 12:58 PM - nobu (Nobuyoshi Nakada)

- Description updated

Another thread may make a substring of the buffer string during writing.
Then the temporarily frozen string becomes a shared root of the new substring.
Later the root gets unfrozen and modified, now the substring has an invalid pointer.

> If another thread calls String#freeze on the string we are
> currently writing; we will blindly unfreeze it during
> fwrite_unfreeze from ensure. However, I do not expect this to
> be a real-world case.

That is test_threaded_flush in test_io.rb,
freezing the same string simultaneously in io_fwrite.

#6 - 01/05/2017 11:09 PM - normalperson (Eric Wong)

- File 0001-v2-io.c-io_fwrite-copy-to-hidden-buffer-when-writing.patch added

OK, different strategy; not as fast, but still better than what
we currently have.

[PATCH v2] io.c (io_fwrite): copy to hidden buffer when writing

This avoids garbage from IO#write for [Bug #13085] when
called in a read-write loop while protecting the VM
from race conditions forced by the user.

Memory usage from benchmark/bm_io_copy_stream_write.rb
is reduced greatly:

target 0: a (ruby 2.5.0dev (2017-01-05 trunk 57270) [x86_64-linux])
target 1: b (ruby 2.5.0dev (2017-01-05) [x86_64-linux])

Memory usage (last size) (B)
name a b
io_copy_stream_write 81899520.000 6561792.000

Memory consuming ratio (size) with the result of `a' (greater is better)
name b
io_copy_stream_write 12.481

Despite the extra deep data copy, there is a small speedup in
execution time due to GC avoidance:

Execution time (sec)
name a b
io_copy_stream_write 0.393 0.296

Speedup ratio: compare with the result of `a' (greater is better)
name b
io_copy_stream_write 1.328

This patch increases memory bandwidth use by pessimistically
copying the data into a temporary hidden buffer. Using a
lightweight frozen copy (as before this patch) is ineffective
in read-write loops, since the read operation will trigger
a heavy copy behind our back due to the CoW operation.

It is also impossible to safely release memory from the
lightweight CoW string, because we have no idea how many
lightweight duplicates exist by the time we reacquire GVL.

So, we now make a heavy copy up front which we recycle
immediately upon completion.

Ideally, Ruby should have a way of detecting Strings which are
not visible to other threads and be able to optimize away the
internal copy. Or, we give up on the idea of implicit data
sharing between threads since its dangerous anyways.

#7 - 01/13/2017 06:31 AM - normalperson (Eric Wong)

normalperson@yhbt.net wrote:

> File 0001-v2-io.c-io_fwrite-copy-to-hidden-buffer-when-writing.patch added
>
> OK, different strategy; not as fast, but still better than what
> we currently have.

Any comments on my alternative patch?

Anyways, I don't like this v2 strategy, either, since it has an
extra memory copy, but it's still better than creating gigantic
amounts of garbage.

With the existence of ObjectSpace.each_object and ability for
threads to be spawned inside signal handlers, I'm not sure if
there's a performant way for us to check the thread-visibility
of a particular String to elide the copy.

> https://bugs.ruby-lang.org/issues/13085#change-62398

#8 - 01/17/2017 12:08 AM - normalperson (Eric Wong)

- File 0001-basicsocket-Linux-workaround-for-excess-garbage-on-w.patch added

At least on Linux, this is probably the best performance we can get
with sockets. Unfortunately, this does not affect writes to pipes
(write_nonblock is OK for me, at least), and regular files
(where we really need to release GVL)

--------8<-------
Subject: [PATCH] basicsocket: Linux workaround for excess garbage on write

Linux allows us to use the MSG_DONTWAIT flag on sockets
to do non-blocking send(2) without side effects of modifying
the underlying file flags. This allows us to replace the
generic IO#write and avoid garbage creation on GVL release:
https://bugs.ruby-lang.org/issues/13085
We now only release the GVL when IO cannot proceed.

While IO#write_nonblock may be used for pipes and sockets
portably, that has the side effect of changing the file flag via
fcntl. MSG_DONTWAIT under Linux has no side effects on the
socket.

I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.
I HATE IMPLICITLY SHARED MEMORY.

#9 - 01/28/2017 08:50 AM - normalperson (Eric Wong)

- File 0001-io.c-recycle-garbage-on-write.patch added

I think this can be a universal solution. Lightly tested and all tests pass,
but I have not checked coverage, yet.

I reuse one of the embed length bits for shared (noembed) string
to track when a string is shared multiple times. In the common
case, there is one share (the original source string), so we can
recycle immediately in ensure. If a second share is set, the
new bit is set and we do not recycle in ensure.

#10 - 01/29/2017 01:02 AM - nobu (Nobuyoshi Nakada)

- Assignee set to normalperson (Eric Wong)

- Status changed from Open to Assigned

Seems nice, let's try it.

#11 - 01/30/2017 08:41 PM - normalperson (Eric Wong)

nobu@ruby-lang.org wrote:

> Seems nice, let's try it.

Thanks, r57469. I'll work on syswrite and send* (socket) next.

#12 - 01/30/2017 09:54 PM - Anonymous

- Status changed from Assigned to Closed

Applied in changeset r57471.

---

string.c (rb_str_tmp_frozen_release): release embedded strings

Handle the embedded case first, since we may have an embedded
duplicate and non-embedded original string.

- string.c (rb_str_tmp_frozen_release): handled embedded strings
- test/ruby/test_io.rb (test_write_no_garbage): new test [ruby-core:78898] [Bug #13085]

#13 - 01/31/2017 02:51 AM - normalperson (Eric Wong)

Eric Wong normalperson@yhbt.net wrote:

> nobu@ruby-lang.org wrote:
>
> > Seems nice, let's try it.
>
> Thanks, r57469. I'll work on syswrite and send* (socket) next.

I guess send* in socket never froze the string. I guess it
will be necessary...

Anyways, I made sprintf and strftime avoid garbage
in a similar cases (r57473 and r57476), too.

Also, I've been thinking the rb_ensure usage for this in io.c
isn't necessary. Exceptions ought to be rare and I already
consider anything which repeatedly throws exceptions a lost
cause. So I think I'll apply the following, tomorrow:

https://80x24.org/spew/20170131024140.27859-1-e@80x24.org/raw

The diffstat alone is seems worth it:

io.c | 93 +++++++++++++++++--
1 file changed, 20 insertions(+), 73 deletions(-)

#14 - 03/12/2017 06:06 PM - naruse (Yui NARUSE)

- Related to Bug #13299: backport r57469, r57472, r57508 (garbage reduction for IO#write/syswrite) added

Files

0001-io.c-io_fwrite-temporarily-freeze-string-when-writin.patch	2.6 KB	12/31/2016	normalperson (Eric Wong)
0001-v2-io.c-io_fwrite-copy-to-hidden-buffer-when-writing.patch	2.9 KB	01/05/2017	normalperson (Eric Wong)
0001-basicsocket-Linux-workaround-for-excess-garbage-on-w.patch	5.54 KB	01/17/2017	normalperson (Eric Wong)
0001-io.c-recycle-garbage-on-write.patch	6.33 KB	01/28/2017	normalperson (Eric Wong)