Ruby - Bug #20863

`zlib.c` calls `rb_str_set_len` and `rb_str_modify_expand`(and others) without holding the GVL.

11/05/2024 10:47 AM - ioquatix (Samuel Williams)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	3.1: UNKNOWN, 3.2: UNKNOWN, 3.3: UNKNOWN

Description

Background

I was working on https://bugs.ruby-lang.org/issues/20876 and was investigating some problems with zlib.c and GVL, and noticed that zstream_run_func is executed without the GVL, but can invoke various rb_ string functions. Those functions in turn can invoke rb_raise and generally look problematic. However, maybe by luck, such code path does not appear to be invoked in typical usage.

However, even so, it is possible to cause zstream_run_func to segfault by a carefully crafted program which causes the internal buffer to be resized while the GVL is released: https://github.com/ruby/zlib/pull/88#issuecomment-2455772054

Proposal

I would like to modify zlib.c to only release the GVL around the CPU intensive compression/decompression operation: https://github.com/ruby/zlib/pull/88

In addition, I identified several more improvements to prevent segfaults and other related failures:

- Use rb_str_locktemp to prevent the z->buf changing size while in use by the rb_nogvl code.
- Expand the mutex to protect #deflate and #inflate completely, not just the internal operation.

In order to catch these issues earlier and find other bugs like this, I recommend we introduce additional checks: https://bugs.ruby-lang.org/issues/20877

Associated revisions

Revision b143fd5bd8527da3ddd176a3d6a362d0ab3bc6c7 - 11/20/2024 09:02 PM - Samuel Williams

[ruby/zlib] Don't call rb_str_set_len while released the GVL.
(https://github.com/ruby/zlib/pull/88)

- Only release the GVL where necessary.

- Several string manipulation methods were invoked while the GVL was released. This is unsafe.
- The mutex protecting multi-threaded access was not covering buffer state manipulation, leading to data corruption and out-of-bounds writes.
- Using rb_str_locktmp prevents changes to buffer while it's in use.

[Bug #20863]

https://github.com/ruby/zlib/commit/e445cf3c80

Revision b143fd5bd8527da3ddd176a3d6a362d0ab3bc6c7 - 11/20/2024 09:02 PM - Samuel Williams

[ruby/zlib] Don't call rb_str_set_len while released the GVL.
(https://github.com/ruby/zlib/pull/88)

- Only release the GVL where necessary.

- Several string manipulation methods were invoked while the GVL was released. This is unsafe.
- The mutex protecting multi-threaded access was not covering buffer state manipulation, leading to data corruption and out-of-bounds writes.
- Using rb_str_locktmp prevents changes to buffer while it's in use.

[Bug #20863]

https://github.com/ruby/zlib/commit/e445cf3c80

Revision b143fd5b - 11/20/2024 09:02 PM - Samuel Williams

[ruby/zlib] Don't call rb_str_set_len while released the GVL.
(https://github.com/ruby/zlib/pull/88)

- Only release the GVL where necessary.

- Several string manipulation methods were invoked while the GVL was released. This is unsafe.
- The mutex protecting multi-threaded access was not covering buffer state manipulation, leading to data corruption and out-of-bounds writes.
- Using rb_str_locktmp prevents changes to buffer while it's in use.

[Bug #20863]

https://github.com/ruby/zlib/commit/e445cf3c80

History

#1 - 11/05/2024 11:02 AM - ioquatix (Samuel Williams)

- Description updated

#2 - 11/06/2024 10:14 PM - ioquatix (Samuel Williams)

- Description updated

#3 - 11/07/2024 12:58 AM - ioquatix (Samuel Williams)

- Description updated

#4 - 11/07/2024 02:50 PM - byroot (Jean Boussier)

@ko1 (Koichi Sasada) Do we have a proper description of what is safe and what it unsafe to do with the GVL released?

Because obviously it's OK to use ruby_xmalloc / ruby_xfree with the GVL released, so methods which allocate aren't necessarily problematic?\

In this case I'm unclear on why rb_str_set_len / rb_str_modify_expand shouldn't be called with the GVL released, assuming the objects on which they operate aren't visible to any other thread.

I think it would be helpful to have more clear guidelines on these things (unless of course I missed some existing documentation).

#5 - 11/07/2024 04:46 PM - ko1 (Koichi Sasada)

Quoted from rb_thread_call_without_gvl doc:

```
* NOTE: You can not execute most of Ruby C API and touch Ruby
*       objects in `func()' and `ubf()', including raising an
*       exception, because current thread doesn't acquire GVL
*       (it causes synchronization problems).  If you need to
*       call ruby functions either use rb_thread_call_with_gvl()
*       or read source code of C APIs and confirm safety by
*       yourself.
*
* NOTE: In short, this API is difficult to use safely.  I recommend you
*       use other ways if you have.  We lack experiences to use this API.
*       Please report your problem related on it.
*
* NOTE: Releasing GVL and re-acquiring GVL may be expensive operations
*       for a short running `func()'. Be sure to benchmark and use this
*       mechanism when `func()' consumes enough time.
*
* Safe C API:
* * rb_thread_interrupted() - check interrupt flag
* * ruby_xmalloc(), ruby_xrealloc(), ruby_xfree() -
*   they will work without GVL, and may acquire GVL when GC is needed.
```

Again:

```
* NOTE: In short, this API is difficult to use safely.  I recommend you
*       use other ways if you have.  We lack experiences to use this API.
*       Please report your problem related on it.
```

#6 - 11/07/2024 05:07 PM - byroot (Jean Boussier)

@ko1 (Koichi Sasada) Not sure how I didn't think to check that, thank you. So indeed allocations are fine. From what I understand, the issue is mostly exceptions and of course using an object concurrently.

#7 - 11/07/2024 08:27 PM - ioquatix (Samuel Williams)

I think the issue is, those methods from a public interface POV, are not allowed to be called without the GVL.

Even if today the implementation follows a "safe" code path, in the future, it may not.

Adding these annotations will help to clarify that "this method is not safe to call without the GVL" - a form of internal and run-time documentation.

#8 - 11/07/2024 09:25 PM - Eregon (Benoit Daloze)

ioquatix (Samuel Williams) wrote in #note-7:

> Even if today the implementation follows a "safe" code path, in the future, it may not.

This is a good point.
I think we should consider all C API functions unsafe to be called without the GVL, except the functions listed in Safe C API.
So I think we should update the docs to remove or read source code of C APIs and confirm safety by yourself. as it's not a good idea as it may change and it's very hard to assess if safe.

#9 - 11/07/2024 09:28 PM - byroot (Jean Boussier)

There would be quite a lot of value in having *some* nogvl save APIs though. e.g. if database clients could allocate Hash/Array/String to build the response while the GVL is still released, it could really help with throughput of threaded servers like Puma.

#10 - 11/07/2024 09:29 PM - ioquatix (Samuel Williams)

> There would be quite a lot of value in having some nogvl save APIs though. e.g. if database clients could allocate Hash/Array/String to build the response while the GVL is still released, it could really help with throughput of threaded servers like Puma.

I think it's a great idea (seriously great), but out of scope for this issue. Do you want to create a new issue to start that discussion?

#11 - 11/20/2024 09:02 PM - Anonymous

- Status changed from Open to Closed

Applied in changeset git|b143fd5bd8527da3ddd176a3d6a362d0ab3bc6c7.

[ruby/zlib] Don't call rb_str_set_len while released the GVL.
(https://github.com/ruby/zlib/pull/88)

- Only release the GVL where necessary.

- Several string manipulation methods were invoked while the GVL was released. This is unsafe.
- The mutex protecting multi-threaded access was not covering buffer state manipulation, leading to data corruption and out-of-bounds writes.
- Using rb_str_locktmp prevents changes to buffer while it's in use.

[Bug #20863]

https://github.com/ruby/zlib/commit/e445cf3c80