Ruby - Bug #19043

Segfault on macOS 11.7 while using StringScanner in multiple threads

10/06/2022 09:08 PM - keithdoggett (Keith Doggett)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 3.2.0dev (2022-09-27T18:58:28Z master 5d4048e0bc) [x86_64-darwin19]	**Backport:**	2.7: UNKNOWN, 3.0: UNKNOWN, 3.1: UNKNOWN

Description

During testing on our CI, one of the runners failed due to a segfault that appears to have originated from the StringScanner class, specifically the scan_until method. The test ensures that we are able to properly parse strings in a multithreaded environment.

```
  def test_multithreaded
    parser = RGeo::WKRep::WKTParser.new
    data = fixtures.join("isere.wkt").read
    Array.new(100) do
      Thread.fork do
        parser.parse(data)
      end
    end.map(&:join)
  end
```

Here's the parse method

```
      def parse(str)
        @mutex.synchronize do
          str = str.downcase
          @cur_factory = @exact_factory
          if @cur_factory
            @cur_factory_support_z = @cur_factory.property(:has_z_coordinate) ? true : false
            @cur_factory_support_m = @cur_factory.property(:has_m_coordinate) ? true : false
          end
          @cur_expect_z = nil
          @cur_expect_m = nil
          @cur_srid = @default_srid
          if @support_ewkt && str =~ /^srid=(\d+);/i
            str = $'
            @cur_srid = Regexp.last_match(1).to_i
          end
          begin
            start_scanner(str)
            obj = parse_type_tag
            if @cur_token && !@ignore_extra_tokens
              raise Error::ParseError, "Extra tokens beginning with #{@cur_token.inspect}."
            end
          ensure
            clean_scanner
          end
          obj
        end
      end
```

Where the StringScanner is created and assigned to @scanner in start_scanner and @scanner is set to nil in clean_scanner. According to the control frame information in the log, the error is caused in the scan_until method, but it might be due to gc_sweep being run at some point.

Unfortunately since this happened on a CI system I don't have access to the diagnostic file. We've tried to replicate this locally unsuccessfully. The best we've done is caused a deadlock while trying to join the threads, but cannot reliably reproduce that. Here's a link to the CI run that caused the issue if that's helpful (https://github.com/rgeo/rgeo/actions/runs/3144578897/jobs/5110771257).

If there's any tips on how to reproduce or anything you want me to try to get more information please let me know.

History

#1 - 10/11/2022 01:19 AM - nobu (Nobuyoshi Nakada)

This seems related to compaction-GC, since crashed at revert_stack_objects.
@tenderlovemaking (Aaron Patterson), any thoughts?

#2 - 10/11/2022 08:32 PM - eightbitraptor (Matt V-H)

keithdoggett (Keith Doggett) wrote:

> If there's any tips on how to reproduce or anything you want me to try to get more information please let me know.

@keithdogget I can see that you run with GC.auto_compact=true on CI (from here).

This looks like it is related to auto-compaction.

```
/Users/runner/.rubies/ruby-head/lib/libruby.3.2.dylib(gc_sweep+0x9f6) [0x108ebac46]
/Users/runner/.rubies/ruby-head/lib/libruby.3.2.dylib(newobj_alloc+0x19f) [0x108eb92cf]
/Users/runner/.rubies/ruby-head/lib/libruby.3.2.dylib(rb_wb_protected_newobj_of+0xab) [0x108eaacbb]
```

GC is being triggered while allocating a new object, running a major and then compacting.

Have you tried replicating with GC.auto_compact=true and GC.stress=true?

#3 - 10/20/2022 09:35 PM - keithdoggett (Keith Doggett)

eightbitraptor (Matthew Valentine-House) wrote in #note-2:

> Have you tried replicating with GC.auto_compact=true and GC.stress=true?

Thanks for the response. We tried to replicate the crash with GC.stress=true but were unable to do so, although we were able to cause a few deadlocks (though we're unsure what's causing it exactly). We even decomposed the method to test just the StringScanner related functionality in a mutex to no avail.

I can keep trying to test it on my end, but the deadlocks seem to randomly happen. Maybe if I can figure out the cause of those that will give us more info on the root cause the crash?

Files

multithread_crash.log	75.3 KB	10/06/2022	keithdoggett (Keith Doggett)