Ruby master - Bug #17220

Rails Active Job integration test fails with Ruby 3.0.0 since 2038cc6cab6ceeffef3ec3a765c70ae684f829ed

10/07/2020 01:26 PM - yahonda (Yasuo Honda)

Status: Closed
Priority: Normal
Assignee: Glass_saga (Masaki Matsushita)
Target version: ruby

Backport: 2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONTNEED

Description
One of the Rails CI, Active Job integration test with sidekiq, against Ruby 3.0.0 has been failing since August 30, 2020.

According to git bisect is it triggered by 2038cc6cab6ceeffef3ec3a765c70ae684f829ed. Somehow this issue only reproduces with Ruby on Docker like rubylang/ruby:master-nightly-bionic. It does not reproduce if Ruby is installed locally using rbenv install 3.0.0-dev on Ubuntu 20.04 and macOS 11 beta.

The first failed build job
https://buildkite.com/rails/rails/builds/71321#84b29655-b3df-4b5c-8b20-cbf15ecd9653

Ruby 2.8.0p-1 (2020-08-29 revision d7492a0be885ea9f2b9f71e3e95582f9a859c439) [x86_64-linux]

The last successful build job
https://buildkite.com/rails/rails/builds/71143#369217f7-95f6-4ab9-8ef5-7c6364bd803e

Ruby 2.8.0p-1 (2020-08-20 revision a74df67244199d1fd1f7a20b49dd5a096d2a13a2) [x86_64-linux]

git bisect result

Performed git bisect ruby/ruby repository and it says 2038cc6cab6ceeffef3ec3a765c70ae684f829ed triggers this build failure.

Steps to reproduce

1. Install Docker
2. Install Ruby 2.7.1 (or whatever Ruby version to run rake)
3. Create Ruby docker image for 2038cc6cab6ceeffef3ec3a765c70ae684f829ed

```
git clone https://github.com/ruby/ruby-docker-images.git
cd ruby-docker-images
rake docker:build ruby_version=master:2038cc6cab6ceeffef3ec3a765c70ae684f829ed
```

1. Run Rails CI using the Docker image created in step 3

```
cd ~
git clone https://github.com/rails/rails.git
cd rails
```

```
git clone https://github.com/rails/buildkite-config .buildkite/
RUBY_IMAGE=rubylang/ruby:master-2038cc6cab6ceeffef3ec3a765c70ae684f829ed-bionic docker-compose -f .buildkite/docker-compose.yml build base & & CI=1 docker-compose -f .buildkite/docker-compose.yml run default runner activejob 'AJ_ADAPTER=sidekiq AJ_INTEGRATION_TESTS=true bin/test test/integration/queuing_test.rb --seed 5170'
```

Actual result
Using sidekiq
Run options: --seed 5170

# Running:

Failure:
QueuingTest#test_should_run_job_enqueued_in_the_future_at_the_specified_time [/rails/activejob/test/integration/queuing_test.rb:76]:
Expected false to be truthy.

```
bin/test test/integration/queuing_test.rb:71
.F
```

Failure:
QueuingTest#test_should_run_jobs_enqueued_on_a_listening_queue [/rails/activejob/test/integration/queuing_test.rb:14]:
Expected false to be truthy.

```
bin/test test/integration/queuing_test.rb:11
.SS..F
```

Failure:
QueuingTest#test_current_locale_is_kept_while_running_perform_later [/rails/activejob/test/integration/queuing_test.rb:102]:
Expected false to be truthy.

```
bin/test test/integration/queuing_test.rb:93
.F
```

Failure:
QueuingTest#test_current_timezone_is_kept_while_running_perform_later [/rails/activejob/test/integration/queuing_test.rb:119]:
Expected false to be truthy.

```
bin/test test/integration/queuing_test.rb:110
.
```

Finished in 34.153644s, 0.4392 runs/s, 0.3514 assertions/s.
15 runs, 12 assertions, 4 failures, 0 errors, 5 skips

You have skipped tests. Run with --verbose for details.

**Expected result**

It should success as the Ruby as of the previous commit 1035a3b202ee86bf2b0a1d00eefcfff0d7ab9f6b does.

```
$ RUBY_IMAGE=rubylang/ruby:master-1035a3b202ee86bf2b0a1d00eefcfff0d7ab9f6b-bionic docker-compose -f ./buildkite/docker-compose.yml build base & & CI=1 docker-compose -f ./buildkite/docker-compose.yml run default_runner activejob 'AJ_ADAPTER=sidekiq AJ_INTEGRATION_TESTS=true bin/test/test/integration/queuing_test.rb --seed 5170'
```

```bash
+++ activejob: AJ_ADAPTER=sidekiq AJ_INTEGRATION_TESTS=true bin/test/test/integration/queuing_test.rb --seed 5170
Using sidekiq
Run options: --seed 5170
```
# Running:
.SSS....SS.....

Finished in 13.647623s, 1.0991 runs/s, 1.0258 assertions/s.
15 runs, 14 assertions, 0 failures, 0 errors, 5 skips

You have skipped tests. Run with --verbose for details.

## Related Issues:
Related to Ruby master - Feature #17134: Add resolv_timeout to TCPSocket

## Associated revisions

### Revision 94d49ed3 - 12/04/2020 02:31 PM - Glass_saga (Masaki Matsushita)
Add a hook before fork() for getaddrinfo_a()

We need stop worker threads in getaddrinfo_a() before fork().
This change adds a hook before fork() that cancel all outstanding requests
and wait for all ongoing requests. Then, it waits for all worker
threads to be finished.

Fixes [Bug #17220]

### Revision 5d8bcc48 - 12/07/2020 04:33 AM - Glass_saga (Masaki Matsushita)
Revert getaddrinfo_a()

getaddrinfo_a() gets stuck after fork().
To avoid this, we need 1 second sleep to wait for internal
worker threads of.getaddrinfo_a() to be finished, but that is unacceptable.

[Bug #17220] [Feature #17134] [Feature #17187]

## History

### #1 - 10/15/2020 11:57 PM - eugeneius (Eugene Kenny)
Here's a small script that reproduces the problem:

```ruby
require "socket"
Socket.getaddrinfo("localhost", nil)
pid = fork do
  Socket.getaddrinfo("localhost", nil)
end
Process.wait pid
```

In Rails' test suite, the fork happens [here](#), and the calls to getaddrinfo happen in [redis-rb](#).
It seems like a lock from the first getaddrinfo is still in place in the forked process; adding sleep 1 before forking prevents the hang.

### #2 - 10/16/2020 10:49 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Third Party's Issue

getaddrinfo_a seems to create a thread, and it will be lost by fork.
I'm not sure if this is a glibc's spec or bug.

This code demonstrates the behavior without Ruby:

```c
#define _GNU_SOURCE
#include <netdb.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/wait.h>

int
```
test_gai(const char *where, const char *name)
{
    printf("in %s: ", where);
    fflush(stdout);
    struct gaicb req = {.ar_name = name};
    struct gaicb *reqs[1] = {&req};
    int ret = getaddrinfo_a(GAI_NOWAIT, reqs, 1, NULL);
    if (ret) goto error;
    ret = gai_suspend((const struct gaicb **)reqs, 1, NULL);
    if (ret) goto error;
    ret = gai_error(reqs[0]);
    if (ret) goto error;
    printf("%s
", name ? name : "noname");
    return 0;
error:
    fprintf(stderr, "failed: ", gai_strerror(ret));
    return ret;
}

int main(int argc, char **argv)
{
    test_gai("parent", "localhost");
    if (argc > 1) {
        printf("sleeping...");
        fflush(stdout);
        usleep((useconds_t)(atof(argv[1])*1e6));
        printf("n");
    }
    pid_t pid = fork();
    if (pid == 0) {
        test_gai("child", "localhost");
        return 0;
    }
    int status;
    waitpid(pid, &status, 0);
    return status;
}

$ gcc -o gaitest gaitest.c -lanl
$ ./gaitest 1
in parent: noname
sleeping...
in child: noname
$ ./gaitest 0.9
in parent: noname
sleeping...
in child: "C"

#3 - 10/26/2020 11:31 PM - eugeneius (Eugene Kenny)
I checked the glibc source, and it intentionally keeps the thread alive:

/* If the runlist is empty, then we sleep for a while, waiting for
something to arrive in it. */
if (runp == NULL && optim.gai_idle_time >= 0)

int gai_idle_time;     /* Number of seconds before idle thread
terminates. */

   gai_idle_time is hardcoded to 1 second, there's no way to configure it.

   When using glibc in a C program, you can call getaddrinfo instead of getaddrinfo_a if you know the program will fork soon. Ruby programs don't have
that option, since getaddrinfo is no longer exposed when getaddrinfo_a is available. I still think this is a bug, although I'm not sure how to solve it
without avoiding getaddrinfo_a and managing the thread directly.

#4 - 10/27/2020 07:28 AM - nobu (Nobuyoshi Nakada)
As getaddrinfo and getaddrinfo_a are not in the async-signal-safe function list in POSIX.1-2001, it can't work in both way.
So the "POSIXLY-correct" answer is "you must not use that method in forked process".

03/15/2022
#5 - 11/24/2020 06:27 AM - yahonda (Yasu Honda)
I also would like to request to reconsider this change. Ruby programmers have no control to use C level getaddrinfo or getaddrinfo_a to avoid deadlocks.

#6 - 11/24/2020 07:07 AM - sawa (Tsuyoshi Sawada)
I see that you have been posting a couple of Rails CI failures as "Ruby bugs" on this site.
I think you are completely mistaken. As far as I know, this website is Ruby's issue tracker, not Rails'.
This issue should be closed as third party issue.

#7 - 11/24/2020 07:32 AM - mame (Yusuke Endoh)
sawa (Tsuyoshi Sawada) wrote in #note-6:

I see that you have been posting a couple of Rails CI failures as "Ruby bugs" on this site.
I think you are completely mistaken. As far as I know, this website is Ruby's issue tracker, not Rails'.

No!! yahonda (Yasu Honda)’s report is super valuable for us. I really appreciate it. Please, please continue.

#8 - 11/24/2020 07:47 AM - naruse (Yui NARUSE)
- Backport changed from 2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN to 2.5: DONTNEED, 2.6: DONTNEED, 2.7: DONTNEED
- Target version set to 3.0
- Assignee set to Glass_saga (Masaki Matsushita)
- Status changed from Third Party’s Issue to Assigned

Thank you for pointing out and sorry that we didn’t treat this issue as not so important.
This issue is considered as blocking issue for Ruby 3.0.
Thank you for reporting and also appreciate about ping us.

#9 - 11/24/2020 08:28 AM - naruse (Yui NARUSE)

As getaddrinfo and getaddrinfo_a are not in the async-signal-safe function list in POSIX.1-2001, it can't work in both way.
So the "POSIXLY-correct" answer is "you must not use that method in forked process".

I think nobu wrote this sentence as "you" means "Ruby users".
But it's not reasonable because it's too hard for Ruby users to ensure running getaddrinfo_a is finished before forking.
Socket.getaddrinfo should ensure finish or kill all their threads and re-run threads if needed in atfork and afterfork.

#10 - 11/24/2020 08:40 AM - sawa (Tsuyoshi Sawada)
mame (Yusuke Endoh) wrote in #note-7:

sawa (Tsuyoshi Sawada) wrote in #note-6:

I see that you have been posting a couple of Rails CI failures as "Ruby bugs" on this site.
I think you are completely mistaken. As far as I know, this website is Ruby's issue tracker, not Rails'.

No!! yahonda (Yasu Honda)’s report is super valuable for us. I really appreciate it. Please, please continue.

I don't get it. If this is a Ruby's bug, then why does this issue only/mainly describe a Rails' error? Even the issue title is about Rails. If a Ruby's bug was found while checking Rails CI, then the latter could be mentioned as a side note, but still, the bug itself could be narrowed down and extracted without mentioning Rails.
If it is a feature request, then it should have been tagged as a feature, not a bug. Then, showing a Rails error may help illustrating a use case.
It is ironical that mame-san has made such a comment because I had thought that, if this "bug report" was made intentionally, just posting a Rails CI error as if it were a Ruby's bug, then it is very rude of the poster against Ruby developers (although I am just a bystander and cannot really say for sure).

#11 - 11/24/2020 09:05 AM - naruse (Yui NARUSE)
sawa (Tsuyoshi Sawada) wrote in #note-10:

I don't get it. If this is a Ruby's bug, then why does this issue only/mainly describe a Rails' error? Even the issue title is about Rails. If a Ruby's bug was found while checking Rails CI, then the latter could be mentioned as a side note, but still, the bug itself could be narrowed down and extracted without mentioning Rails.

We are practical.
In this case yahonda report this ticket, and eugeneius identify the small reproducible code.
Now it's clear that Ruby has an issue.
What is a problem?
You dogmatic attitude doesn't contribute anything.
Contribution is welcome.
What is a contribution is decided by whether it contributes something or not.

#12 - 11/24/2020 09:39 AM - sawa (Tsuyoshi Sawada)
naruse (Yui NARUSE) wrote in #note-11:

In this case yahonda report this ticket, and eugeneius identify the small reproducible code.

That discussion should have been done in Rails' issue tracker.
Now it's clear that Ruby has an issue.

Only after this point would it make sense to report it as a Ruby bug, mentioning eugeneius' small reproducible code.
You dogmatic attitude doesn't contribute anything.
Contribution is welcome.
What is a contribution is decided by whether it contributes something or not.

According to naruse-san's comment, the original post did not make any contribution to Ruby. The contribution was made by eugeneius.
You may say that the post triggered a conversation that led another person to find a Ruby's bug. That is indirect still. If that was the intention of the post, the description could have at least been something like "I could not narrow down the issue, but I can tell from observing Rails CI that there must be something wrong with Ruby. Blah blah ..." Just posting a Rails error as a Ruby bug does not make sense unless you have a dogma that any change in Ruby's behavior that makes Rails go wrong is a Ruby's bug. That is perhaps the attitude under which the comment https://bugs.ruby-lang.org/issues/10845#change-88690 was made.

#13 - 11/24/2020 09:47 AM - naruse (Yui NARUSE)
sawa (Tsuyoshi Sawada) wrote in #note-12:

naruse (Yui NARUSE) wrote in #note-11:

In this case yahonda report this ticket, and eugeneius identify the small reproducible code.

That discussion should have been done in Rails' issue tracker.

I need this issue as Release manager of Ruby 3.0.
I don't need further your comment.
If you continue this in this thread, I'll block you.

#14 - 11/24/2020 11:00 AM - mame (Yusuke Endoh)
"A new version of Ruby breaks an existing application" is one of the most valuable reports. Especially if the application is popular and practical like Rails.
yahonda (Yasuo Honda) has reported regression bugs of Ruby, so his reports are relatively trustable and he is an important contributor for us.
Even if the breakage is caused by an intentional change, it is still informative to estimate the impact of the incompatibility.
We may eventually reject such reports, but we would really appreciate them.
Small reproducible code is helpful for us, but not mandatory for bug reports. It is the worst possible thing that people hesitate to report a regression bug in a real-world application just because they have no time to create small reproducible code. Other people may be able to create such code (like eugeneius (Eugene Kenny) in this ticket). We cannot promise but we may identify and fix an issue without code. Even if we couldn't determine if it is a regression or not, it may be useful in future to keep a record.

#15 - 11/25/2020 12:53 AM - shyouhei (Shyouhei Urabe)
Thanks yahonda (Yasuo Honda) for reporting this!

Glass_saga (Masaki Matsushita) Any idea how to fix this? Or revert?

#16 - 11/25/2020 08:04 AM - Glass_saga (Masaki Matsushita)
Thank you for reporting this behavior. yahonda (Yasuo Honda)
I'd like to implement a hook before fork() to cancel all outstanding requests of getaddrinfo_a() and wait for ongoing requests to be finished.

#17 - 11/25/2020 09:05 AM - naruse (Yui NARUSE)
As glass_saga says,

- Before fork, all pthreads need to be stopped (this is considered Unix's practical restriction as far as I understand. see Bug #2724 as an example)
- getaddrinfo_a uses their own pthread to provide async feature. It has a queue and pthreads (up to 20 threads) to handle DNS requests.
- getaddrinfo_a with NOWAIT posts a task to the queue.
- With gai_cancel(3), it can remove queued task. But it doesn't remove/stop already running task.
- With gai_error(3), it can get the status of the task.
- To stop those pthreads, it needs to stop posting new task, remove all queued tasks, and then stop/wait running tasks. After that worker threads will finish after 1 second sleep. To skip that sleep, we need to call an internal glibc API __gai_new_request_notification().
- To stop posting new task, just getting GVL.
- To remove all queued tasks, we can use gai_cancel. Though the manpage says gai_cancel(NULL) removes all queued tasks, such feature is not implemented and actually need to pass all posted requests one by one.
- There's no way to stop running tasks with glibc's getaddrinfo_a (because pthread_t of the worker pthreads are hidden). To stop them, we need to re-implement getaddrinfo_a. With our own impl, we can use pthread_cancel(3) to stop getaddrinfo(3) in the threads. It seems that some issues are found before but actually we can do that. (ref. https://bugzilla.redhat.com/show_bug.cgi?id=1209433)
- In Ruby 3.0 we'll ensure to stop pthreads before fork. In the future, we'll provide further enhancements.

#18 - 11/27/2020 09:22 AM - Glass_saga (Masaki Matsushita)
- File fix_bug17220.patch added

I wrote a patch to fix this bug.
Before fork(), it cancels requests and waits for worker threads of getaddrinfo_a(3) to be finished.
naruse (Yui NARUSE) -san, could you review this?

#19 - 11/27/2020 11:19 AM - Glass_saga (Masaki Matsushita)
- File deleted (fix_bug17220.patch)

#20 - 11/27/2020 11:20 AM - Glass_saga (Masaki Matsushita)
- File fix_bug17220.patch added

patch updated

#21 - 11/27/2020 12:34 PM - nobu (Nobuyoshi Nakada)
rb_rescue(call_getaddrinfo_a_before_exec, Qnil, NULL, Qnil) should be rb_protect?

#22 - 11/27/2020 03:14 PM - Glass_saga (Masaki Matsushita)
- File fix_bug17220_2.patch added

patch updated:

- call rb_getaddrinfo_a_before_exec() before rb_thread_stop_timer_thread()
- use rb_protect() instead of rb_rescue

#23 - 12/02/2020 03:20 PM - Glass_saga (Masaki Matsushita)
- File fix_bug17220_3.patch added

patch updated:
to stop posting new task, we need keep holding GVL.
use function pointer instead of rb_funcall (this may release GVL)
use sleep() instead of rb_thread_sleep() (ditto)

#24 - 12/03/2020 09:41 AM - nobu (Nobuyoshi Nakada)
About __getaddrinfo_a_before_exec_func:

- the name starting with __ is reserved by C standard
- the name seems too specific and can be more generic w/o getaddrinfo_a
- it should not be exposed in public headers

About the test:

- maybe should be under test/socket

Misc:

- 2nd and 4th hunks of ext/socket/raddrinfo.c, and 2nd of process.c seem useless

#25 - 12/04/2020 03:55 AM - Glass_saga (Masaki Matsushita)
- File fix_bug17220_4.patch added

patch updated:

- use more generic name rb_socket_before_exec_func
- move test to test/socket
- remove useless lines

#26 - 12/04/2020 04:05 AM - Glass_saga (Masaki Matsushita)
- File fix_bug17220_cond.patch added

another patch that uses pthread_cond_signal(&__gai_new_request_notification), but it doesn't work.

#27 - 12/04/2020 04:52 AM - nobu (Nobuyoshi Nakada)
Glass_saga (Masaki Matsushita) wrote in #note-25:

patch updated:

- use more generic name rb_socket_before_exec_func

I think that ruby core should just provide the hook always.
It is the responsibility of ext/socket to register the hook or not.

#28 - 12/04/2020 02:33 PM - Glass_saga (Masaki Matsushita)
- Status changed from Assigned to Closed

Applied in changeset git|94d49ed31c39002335eeee65d42463139f561954.

Add a hook before fork() for getaddrinfo_a()

We need stop worker threads in getaddrinfo_a() before fork().
This change adds a hook before fork() that cancel all outstanding requests
and wait for all ongoing requests. Then, it waits for all worker
threads to be finished.

Fixes [Bug #17220]

#29 - 12/06/2020 02:33 PM - yahonda (Yasuo Honda)
Thanks for providing the fix. I have opened a pull request to revert 'sleep 1' workarounds https://github.com/rails/rails/pull/40754.

#30 - 12/08/2020 02:52 AM - nobu (Nobuyoshi Nakada)
- Status changed from Closed to Open

#31 - 12/10/2020 09:06 AM - ko1 (Koichi Sasada)
- Target version deleted (3.0)

#32 - 12/10/2020 09:11 AM - naruse (Yui NARUSE)
- Related to Feature #17134: Add resolv_timeout to TCPSocket added

#33 - 12/27/2020 09:25 AM - naruse (Yui NARUSE)
- Status changed from Open to Closed

Files

<table>
<thead>
<tr>
<th>File Name</th>
<th>Size</th>
<th>Date</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>fix_bug17220.patch</td>
<td>7.32 KB</td>
<td>11/27/2020</td>
<td>Glass_saga (Masaki Matsushita)</td>
</tr>
<tr>
<td>fix_bug17220_2.patch</td>
<td>7.53 KB</td>
<td>11/27/2020</td>
<td>Glass_saga (Masaki Matsushita)</td>
</tr>
<tr>
<td>fix_bug17220_3.patch</td>
<td>7.11 KB</td>
<td>12/02/2020</td>
<td>Glass_saga (Masaki Matsushita)</td>
</tr>
<tr>
<td>fix_bug17220_4.patch</td>
<td>6.64 KB</td>
<td>12/04/2020</td>
<td>Glass_saga (Masaki Matsushita)</td>
</tr>
<tr>
<td>fix_bug17220_cond.patch</td>
<td>7.35 KB</td>
<td>12/04/2020</td>
<td>Glass_saga (Masaki Matsushita)</td>
</tr>
</tbody>
</table>