Ruby - Bug #4463

[PATCH] release GVL for fcntl() for operations that may block

03/03/2011 06:58 PM - normalperson (Eric Wong)

Status:	Closed		
Priority:	Normal		
Assignee:			
Target version:	2.0.0		
ruby -v:	-	**Backport:**	

Description

=begin
Users of F_SETLKW may block the entire VM via IO#fcntl,
release the GVL so other operations may continue.

=end

Associated revisions

Revision c0359f81 - 03/04/2011 04:38 PM - kosaki (Motohiro KOSAKI)

- io.c (io_cntl, nogvl_io_cntl): IO.fcntl() and IO.ioctl()
  release GVL during calling kernel interface.
  Suggested by Eric Wong. [ruby-core:35417][Bug #4463]

- test/ruby/test_io.rb (TestIO#test_fcntl_lock): add new test for
  IO.fcntl().

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@31025 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 03/04/2011 12:23 AM - kosaki (Motohiro KOSAKI)

- ruby -v changed from ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux] to -

=begin

> Issue #4463 has been reported by Eric Wong.
>
> ---
>
> Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
> http://redmine.ruby-lang.org/issues/4463
>
> Author: Eric Wong
> Status: Open
> Priority: Normal
> Assignee:
> Category: core
> Target version: 1.9.x
> ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]
>
> Users of F_SETLKW may block the entire VM via IO#fcntl,
> release the GVL so other operations may continue.

Yeah.
It looks reasonable request. :)
=end

#2 - 03/05/2011 02:24 AM - kosaki (Motohiro KOSAKI)

=begin
Hi

2011/3/3 KOSAKI Motohiro kosaki.motohiro@gmail.com:

> Issue #4463 has been reported by Eric Wong.
>
> ---
>
> Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
> http://redmine.ruby-lang.org/issues/4463
>
> Author: Eric Wong
> Status: Open
> Priority: Normal
> Assignee:
> Category: core
> Target version: 1.9.x
> ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]
>
> Users of F_SETLKW may block the entire VM via IO#fcntl,
> release the GVL so other operations may continue.

Yeah.
It looks reasonable request. :)

Hi

I've commited slightly modified version today (r31025).
The difference is,

1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
   A) if a user are using network filesystem, almost all fcntl need network
   communication. iow, they can be blocked.
   B) We are sure ioctl() has similar issue. But, we don't have any knowledge
   which ioctl can be blocked. It is strongly dependend a
   platform and a device.
2. Added small test. It is based on your Fcntl::Flock patch.

Thanks.
=end

#3 - 03/05/2011 02:24 AM - normalperson (Eric Wong)

=begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

> Hi
>
> I've commited slightly modified version today (r31025).
> The difference is,
>
> 1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
>   A) if a user are using network filesystem, almost all fcntl need network
>   communication. iow, they can be blocked.
>   B) We are sure ioctl() has similar issue. But, we don't have any knowledge
>   which ioctl can be blocked. It is strongly dependend a
>   platform and a device.

Agreed on both points.

> 1. Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? I'd be happy to make
any changes/improvements necessary (+docs, too). Thanks again.

--
Eric Wong
=end

#4 - 03/05/2011 06:23 PM - kosaki (Motohiro KOSAKI)

=begin
Hi

I've commited slightly modified version today (r31025).
The difference is,

1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
   Â Â A) if a user are using network filesystem, almost all fcntl need network
   Â Â Â Â communication. iow, they can be blocked.
   Â Â B) We are sure ioctl() has similar issue. But, we don't have any knowledge
   Â Â Â Â which ioctl can be blocked. It is strongly dependend a
   platform and a device.

Agreed on both points.

thank you.

1. Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? Â I'd be happy to make
any changes/improvements necessary (+docs, too). Â Thanks again.

Umm..
I don't like its interface so much. your flock object don't mange any lock
state. it's merely wrapper of argument of fcntl. your interface mean we need
two line every lock operation. eg.

```
   lock
```

=end

#5 - 03/05/2011 07:23 PM - normalperson (Eric Wong)

=begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

Umm..
I don't like its interface so much. your flock object don't mange any lock
state. it's merely wrapper of argument of fcntl. your interface mean we need
two line every lock operation. eg.

```
  lock = Fcntl::Flock.new Fcntl::F_WRLCK
  f.fcntl Fcntl::F_SETLKW, lock
```

I agree it's currently too verbose.

I tried to keep io.c the same so I used a String subclass. Maybe I
should just modify teach io.c to deal with Hash/Array arguments? I do
worry about placing more burden on io.c for portability reasons, though
POSIX file locks might be very common by now...

To shorten interface, maybe Fcntl::Flock[] can return an array for splat
and take symbol args (like new Socket):

```
f.fcntl *Fcntl::Flock[:F_SETLKW, :F_WRLCK, :SEEK_SET, 0, 0]
```

Or maybe even:

```
f.fcntl *Fcntl::Flock[:SETLKW, :WRLCK, :SET, 0, 0]
```

but I *personally* prefer array or hash capsulation. e.g

f.fcntl Fcntl:F_SETLKW, [Fcntl:F_WRLK, SEEK_SET, 0, 0]

or

f.fcntl Fcntl:F_SETLKW, { :l_type => Fcntl:F_WRLK }

Yes, I like the Hash one but requires modifying io.c with potentially
unportable code to support.

If we use non-String, maybe just call fcntl(2) inside ext/fcntl/fcntl.c

internally and forget about IO#fcntl in io.c entirely:

```
Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)
```

Or even:

```
Fcntl.lock(io, :WRLCK, :SET, 0, 0)
Fcntl.try_lock(io, :WRLCK, :SET, 0, 0)
Fcntl.unlock(io, :SET, 0, 0)
Fcntl.getlock(io, :RDLCK, :SET, 0, 0) -> Fcntl::Flock object
```

That would allow us to do something stateful like:

```
Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end
```

I dislike all caps, even, taking hints from pthread_rwlock_*:

```
Fcntl.rdlock(io, :set, 0, 0)
Fcntl.tryrdlock(io, :set, 0, 0)
Fcntl.wrlock(io, :set, 0, 0)
Fcntl.trywrlock(io, :set, 0, 0)
Fcntl.unlock(io, :set, 0, 0)

Fcntl.read_synchronize(io, :set, 0, 0) do
  # ...
end

Fcntl.synchronize(io, :set, 0, 0) do
  # ...
end
```

> But, of cource, I'm not against if matz ack yours. So I recommend you
> describe the detailed interface to matz instead only just attached a patch.
> It's best practice to persuade *very* busy person. :)

Thanks again for the feedback. So many ways to do this interface,
but just anything but Array#pack sounds good to me :)

--
Eric Wong
=end

#6 - 03/06/2011 10:58 PM - kosaki (Motohiro KOSAKI)

- Status changed from Open to Closed

=begin

=end

#7 - 03/15/2011 05:23 AM - normalperson (Eric Wong)

=begin
Eric Wong normalperson@yhbt.net wrote:

> KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:
>
> > Umm..
> > I don't like its interface so much. your flock object don't mange any lock
> > state. it's merely wrapper of argument of fcntl. your interface mean we need
> > two line every lock operation. eg.
>
> That would allow us to do something stateful like:
>
> Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
>
> ...

end

Following up, I went with something along these lines here.

http://redmine.ruby-lang.org/issues/4464
http://redmine.ruby-lang.org/attachments/1540/0001-add-Fcntl-Lock-object-for-easier-use-of-POSIX-file-l.patch

Simple use case to lock the whole file is just:

Fcntl::Lock.synchronize(file) do
...
end

--
Eric Wong
=end

#8 - 04/12/2011 08:17 PM - kosaki (Motohiro KOSAKI)

=begin

Issue #4463 has been reported by Eric Wong.

Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
http://redmine.ruby-lang.org/issues/4463

Author: Eric Wong
Status: Open
Priority: Normal
Assignee:
Category: core
Target version: 1.9.x
ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]

Users of F_SETLKW may block the entire VM via IO#fcntl,
release the GVL so other operations may continue.

Yeah.
It looks reasonable request. :)
=end

#9 - 04/12/2011 08:17 PM - kosaki (Motohiro KOSAKI)

=begin
Hi

2011/3/3 KOSAKI Motohiro kosaki.motohiro@gmail.com:

> Issue #4463 has been reported by Eric Wong.
>
> ---
>
> Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
> http://redmine.ruby-lang.org/issues/4463
>
> Author: Eric Wong
> Status: Open
> Priority: Normal
> Assignee:
> Category: core
> Target version: 1.9.x
> ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]
>
> Users of F_SETLKW may block the entire VM via IO#fcntl,
> release the GVL so other operations may continue.

Yeah.
It looks reasonable request. :)

Hi

I've commited slightly modified version today (r31025).
The difference is,

1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
   A) if a user are using network filesystem, almost all fcntl need network
   communication. iow, they can be blocked.
   B) We are sure ioctl() has similar issue. But, we don't have any knowledge
   which ioctl can be blocked. It is strongly dependend a
   platform and a device.
2. Added small test. It is based on your Fcntl::Flock patch.

Thanks.
=end

#10 - 04/12/2011 08:17 PM - normalperson (Eric Wong)

=begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

> Hi
>
> I've commited slightly modified version today (r31025).
> The difference is,
>
> 1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
>   A) if a user are using network filesystem, almost all fcntl need network
>   communication. iow, they can be blocked.
>   B) We are sure ioctl() has similar issue. But, we don't have any knowledge
>   which ioctl can be blocked. It is strongly dependend a
>   platform and a device.

Agreed on both points.

> 1. Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? I'd be happy to make
any changes/improvements necessary (+docs, too). Thanks again.

--
Eric Wong
=end

#11 - 04/12/2011 08:17 PM - kosaki (Motohiro KOSAKI)

=begin
Hi

> I've commited slightly modified version today (r31025).
> The difference is,
>
> 1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
>   Â Â A) if a user are using network filesystem, almost all fcntl need network
>   Â Â Â Â communication. iow, they can be blocked.
>   Â Â B) We are sure ioctl() has similar issue. But, we don't have any knowledge
>   Â Â Â Â which ioctl can be blocked. It is strongly dependend a
>   platform and a device.
>
> Agreed on both points.

thank you.

> 1. Added small test. It is based on your Fcntl::Flock patch.
>
> Any chance of that patch making it into trunk? Â I'd be happy to make
> any changes/improvements necessary (+docs, too). Â Thanks again.

Umm..
I don't like its interface so much. your flock object don't mange any lock
state. it's merely wrapper of argument of fcntl. your interface mean we need

two line every lock operation. eg.

```
  lock
```

=end

#12 - 04/12/2011 08:17 PM - normalperson (Eric Wong)

=begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

> Umm..
> I don't like its interface so much. your flock object don't mange any lock
> state. it's merely wrapper of argument of fcntl. your interface mean we need
> two line every lock operation. eg.
>
> ```
>  lock = Fcntl::Flock.new Fcntl::F_WRLCK
>  f.fcntl Fcntl::F_SETLKW, lock
> ```

I agree it's currently too verbose.

I tried to keep io.c the same so I used a String subclass. Maybe I
should just modify teach io.c to deal with Hash/Array arguments? I do
worry about placing more burden on io.c for portability reasons, though
POSIX file locks might be very common by now...

To shorten interface, maybe Fcntl::Flock[] can return an array for splat
and take symbol args (like new Socket):

```
f.fcntl *Fcntl::Flock[:F_SETLKW, :F_WRLCK, :SEEK_SET, 0, 0]
```

Or maybe even:

```
f.fcntl *Fcntl::Flock[:SETLKW, :WRLCK, :SET, 0, 0]
```

> but I *personally* prefer array or hash capsulation. e.g
>
> f.fcntl Fcntl:F_SETLKW, [Fcntl:F_WRLK, SEEK_SET, 0, 0]
>
> or
>
> f.fcntl Fcntl:F_SETLKW, { :l_type => Fcntl:F_WRLK }

Yes, I like the Hash one but requires modifying io.c with potentially
unportable code to support.

If we use non-String, maybe just call fcntl(2) inside ext/fcntl/fcntl.c
internally and forget about IO#fcntl in io.c entirely:

```
Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)
```

Or even:

```
Fcntl.lock(io, :WRLCK, :SET, 0, 0)
Fcntl.try_lock(io, :WRLCK, :SET, 0, 0)
Fcntl.unlock(io, :SET, 0, 0)
Fcntl.getlock(io, :RDLCK, :SET, 0, 0) -> Fcntl::Flock object
```

That would allow us to do something stateful like:

```
Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end
```

I dislike all caps, even, taking hints from pthread_rwlock_*:

```
Fcntl.rdlock(io, :set, 0, 0)
Fcntl.tryrdlock(io, :set, 0, 0)
Fcntl.wrlock(io, :set, 0, 0)
Fcntl.trywrlock(io, :set, 0, 0)
Fcntl.unlock(io, :set, 0, 0)
```

```
Fcntl.read_synchronize(io, :set, 0, 0) do
  # ...
end

Fcntl.synchronize(io, :set, 0, 0) do
  # ...
end
```

> But, of cource, I'm not against if matz ack yours. So I recommend you
> describe the detailed interface to matz instead only just attached a patch.
> It's best practice to persuade *very* busy person. :)

Thanks again for the feedback. So many ways to do this interface,
but just anything but Array#pack sounds good to me :)

--
Eric Wong
=end

#13 - 04/12/2011 08:17 PM - normalperson (Eric Wong)

=begin
Eric Wong normalperson@yhbt.net wrote:

> KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:
>
> > Umm..
> > I don't like its interface so much. your flock object don't mange any lock
> > state. it's merely wrapper of argument of fcntl. your interface mean we need
> > two line every lock operation. eg.
>
> That would allow us to do something stateful like:
>
> Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
>
> **...**
>
> end

Following up, I went with something along these lines here.

http://redmine.ruby-lang.org/issues/4464
http://redmine.ruby-lang.org/attachments/1540/0001-add-Fcntl-Lock-object-for-easier-use-of-POSIX-file-l.patch

Simple use case to lock the whole file is just:

Fcntl::Lock.synchronize(file) do
...
end

--
Eric Wong
=end

#14 - 04/12/2011 08:18 PM - kosaki (Motohiro KOSAKI)

=begin

> Issue #4463 has been reported by Eric Wong.
>
> ---
>
> Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
> http://redmine.ruby-lang.org/issues/4463
>
> Author: Eric Wong
> Status: Open
> Priority: Normal
> Assignee:
> Category: core
> Target version: 1.9.x

> ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]
>
> Users of F_SETLKW may block the entire VM via IO#fcntl,
> release the GVL so other operations may continue.

Yeah.
It looks reasonable request. :)
=end

#15 - 04/12/2011 08:18 PM - kosaki (Motohiro KOSAKI)

=begin
Hi

2011/3/3 KOSAKI Motohiro kosaki.motohiro@gmail.com:

> Issue #4463 has been reported by Eric Wong.
>
> ---
>
> Bug #4463: [PATCH] release GVL for fcntl() for operations that may block
> http://redmine.ruby-lang.org/issues/4463
>
> Author: Eric Wong
> Status: Open
> Priority: Normal
> Assignee:
> Category: core
> Target version: 1.9.x
> ruby -v: ruby 1.9.3dev (2011-03-03 trunk 31011) [x86_64-linux]
>
> Users of F_SETLKW may block the entire VM via IO#fcntl,
> release the GVL so other operations may continue.

Yeah.
It looks reasonable request. :)

Hi

I've commited slightly modified version today (r31025).
The difference is,

1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
   A) if a user are using network filesystem, almost all fcntl need network
   communication. iow, they can be blocked.
   B) We are sure ioctl() has similar issue. But, we don't have any knowledge
   which ioctl can be blocked. It is strongly dependend a
   platform and a device.
2. Added small test. It is based on your Fcntl::Flock patch.

Thanks.
=end

#16 - 04/12/2011 08:18 PM - normalperson (Eric Wong)

=begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

> Hi
>
> I've commited slightly modified version today (r31025).
> The difference is,
>
> 1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
>   A) if a user are using network filesystem, almost all fcntl need network
>   communication. iow, they can be blocked.
>   B) We are sure ioctl() has similar issue. But, we don't have any knowledge
>   which ioctl can be blocked. It is strongly dependend a
>   platform and a device.

Agreed on both points.

1. Added small test. It is based on your Fcntl::Flock patch.

Any chance of that patch making it into trunk? I'd be happy to make
any changes/improvements necessary (+docs, too). Thanks again.

--
Eric Wong
=end

#17 - 04/12/2011 08:18 PM - kosaki (Motohiro KOSAKI)

=begin
Hi

> I've commited slightly modified version today (r31025).
> The difference is,
>
> 1. All IO.fcntl() and IO.iocntl() relese GVL instead only SETLCKW. because,
> Â Â A) if a user are using network filesystem, almost all fcntl need network
> Â Â Â Â communication. iow, they can be blocked.
> Â Â B) We are sure ioctl() has similar issue. But, we don't have any knowledge
> Â Â Â Â which ioctl can be blocked. It is strongly dependend a
> platform and a device.
>
> Agreed on both points.

thank you.

> 1. Added small test. It is based on your Fcntl::Flock patch.
>
> Any chance of that patch making it into trunk? Â I'd be happy to make
> any changes/improvements necessary (+docs, too). Â Thanks again.

Umm..
I don't like its interface so much. your flock object don't mange any lock
state. it's merely wrapper of argument of fcntl. your interface mean we need
two line every lock operation. eg.

```
  lock
```

=end

#18 - 04/12/2011 08:18 PM - normalperson (Eric Wong)

=begin
KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:

> Umm..
> I don't like its interface so much. your flock object don't mange any lock
> state. it's merely wrapper of argument of fcntl. your interface mean we need
> two line every lock operation. eg.
>
> ```
>  lock = Fcntl::Flock.new Fcntl::F_WRLCK
>  f.fcntl Fcntl::F_SETLKW, lock
> ```

I agree it's currently too verbose.

I tried to keep io.c the same so I used a String subclass. Maybe I
should just modify teach io.c to deal with Hash/Array arguments? I do
worry about placing more burden on io.c for portability reasons, though
POSIX file locks might be very common by now...

To shorten interface, maybe Fcntl::Flock[] can return an array for splat
and take symbol args (like new Socket):

```
f.fcntl *Fcntl::Flock[:F_SETLKW, :F_WRLCK, :SEEK_SET, 0, 0]
```

Or maybe even:

```
f.fcntl *Fcntl::Flock[:SETLKW, :WRLCK, :SET, 0, 0]
```

> but I *personally* prefer array or hash capsulation. e.g
>
> f.fcntl Fcntl:F_SETLKW, [Fcntl:F_WRLK, SEEK_SET, 0, 0]
>
> or
>
> f.fcntl Fcntl:F_SETLKW, { :l_type => Fcntl:F_WRLK }

Yes, I like the Hash one but requires modifying io.c with potentially
unportable code to support.

If we use non-String, maybe just call fcntl(2) inside ext/fcntl/fcntl.c
internally and forget about IO#fcntl in io.c entirely:

```
Fcntl::Flock[:WRLCK, :SET, 0, 0].lock(io)
Fcntl::Flock[:WRLCK, :SET, 0, 0].try_lock(io)
Fcntl::Flock[:SET, 0, 0].unlock(io)
```

Or even:

```
Fcntl.lock(io, :WRLCK, :SET, 0, 0)
Fcntl.try_lock(io, :WRLCK, :SET, 0, 0)
Fcntl.unlock(io, :SET, 0, 0)
Fcntl.getlock(io, :RDLCK, :SET, 0, 0) -> Fcntl::Flock object
```

That would allow us to do something stateful like:

```
Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do
  # ...
end
```

I dislike all caps, even, taking hints from pthread_rwlock_*:

```
Fcntl.rdlock(io, :set, 0, 0)
Fcntl.tryrdlock(io, :set, 0, 0)
Fcntl.wrlock(io, :set, 0, 0)
Fcntl.trywrlock(io, :set, 0, 0)
Fcntl.unlock(io, :set, 0, 0)

Fcntl.read_synchronize(io, :set, 0, 0) do
  # ...
end

Fcntl.synchronize(io, :set, 0, 0) do
  # ...
end
```

> But, of cource, I'm not against if matz ack yours. So I recommend you
> describe the detailed interface to matz instead only just attached a patch.
> It's best practice to persuade *very* busy person. :)

Thanks again for the feedback. So many ways to do this interface,
but just anything but Array#pack sounds good to me :)

--
Eric Wong
=end

#19 - 04/12/2011 08:18 PM - normalperson (Eric Wong)

=begin
Eric Wong normalperson@yhbt.net wrote:

> KOSAKI Motohiro kosaki.motohiro@gmail.com wrote:
>
> > Umm..
> > I don't like its interface so much. your flock object don't mange any lock
> > state. it's merely wrapper of argument of fcntl. your interface mean we need
> > two line every lock operation. eg.

That would allow us to do something stateful like:

Fcntl.synchronize(io, :WRLCK, :SET, 0, 0) do

...

end

Following up, I went with something along these lines here.

http://redmine.ruby-lang.org/issues/4464
http://redmine.ruby-lang.org/attachments/1540/0001-add-Fcntl-Lock-object-for-easier-use-of-POSIX-file-l.patch

Simple use case to lock the whole file is just:

```
Fcntl::Lock.synchronize(file) do
# ...
end
```

--
Eric Wong
=end

Files

0001-release-GVL-for-fcntl-for-operations-that-may-block.patch	1.7 KB	03/03/2011	normalperson (Eric Wong)