Ruby trunk - Feature #6615

Release GVL in zlib when calling inflate() or deflate()

06/21/2012 09:20 AM - drbrain (Eric Hodel)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:	2.0.0	

Description

This patch switches from zstream_run from using rb_thread_schedule() to rb_thread_blocking_region().

I don't see a way to safely interrupt deflate() or inflate() so the unblocking function is empty.

This patch should allow use of output buffer sizes larger than 16KB. I suspect 16KB was chosen to allow reasonable context-switching time for ruby 1.8 and earlier. A larger buffer size would reduce GVL contention when processing large streams.

An alternate way to reduce GVL contention would be to move zstream_run's loop outside the GVL, but some manual allocation would be required as currently the loop uses a ruby String as the output buffer.

Associated revisions

Revision 802c468f - 07/02/2012 09:03 PM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run): Process zlib streams without GVL. [Feature #6615]
- NEWS: ditto.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@36270 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 36270 - 07/02/2012 09:03 PM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run): Process zlib streams without GVL. [Feature #6615]
- NEWS: ditto.

Revision 36270 - 07/02/2012 09:03 PM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run): Process zlib streams without GVL. [Feature #6615]
- NEWS: ditto.

Revision 36270 - 07/02/2012 09:03 PM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run): Process zlib streams without GVL. [Feature #6615]
- NEWS: ditto.

Revision 36270 - 07/02/2012 09:03 PM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run): Process zlib streams without GVL. [Feature #6615]
- NEWS: ditto.

Revision 36270 - 07/02/2012 09:03 PM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run): Process zlib streams without GVL. [Feature #6615]
- NEWS: ditto.

Revision 1884f1c0 - 07/03/2012 05:52 AM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run_func): Fix bug that caused early exit of GVL-free loop. [Feature #6615]
- ext/zlib/zlib.c: Fix style to match existing functions.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@36283 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 36283 - 07/03/2012 05:52 AM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run_func): Fix bug that caused early exit of GVL-free loop. [Feature #6615]
- ext/zlib/zlib.c: Fix style to match existing functions.

Revision 36283 - 07/03/2012 05:52 AM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run_func): Fix bug that caused early exit of GVL-free loop. [Feature #6615]
- ext/zlib/zlib.c: Fix style to match existing functions.

Revision 36283 - 07/03/2012 05:52 AM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run_func): Fix bug that caused early exit of GVL-free loop. [Feature #6615]
- ext/zlib/zlib.c: Fix style to match existing functions.

Revision 36283 - 07/03/2012 05:52 AM - drbrain (Eric Hodel)

- ext/zlib/zlib.c (zstream_run_func): Fix bug that caused early exit of GVL-free loop. [Feature #6615]
- ext/zlib/zlib.c: Fix style to match existing functions.

History

#1 - 06/21/2012 11:53 AM - normalperson (Eric Wong)

"drbrain (Eric Hodel)" drbrain@segment7.net wrote:

> I don't see a way to safely interrupt deflate() or inflate() so the
> unblocking function is empty.

I think you can safely specify NULL for the ubf in this case instead of using an empty function.

> This patch should allow use of output buffer sizes larger than 16KB.
> I suspect 16KB was chosen to allow reasonable context-switching time
> for ruby 1.8 and earlier. A larger buffer size would reduce GVL
> contention when processing large streams.

Cool!

#2 - 06/21/2012 03:56 PM - drbrain (Eric Hodel)

- File zlib.release_gvl.2.patch added

This second patch moves the entire run loop outside the GVL, but messes with the internals of String to expand z->buf.

This patch allows large streams to be processed without GVL contention.

#3 - 06/26/2012 08:27 AM - drbrain (Eric Hodel)

- File zlib.release_gvl.3.patch added

This patch allocates one extra byte for the '\0' sentinel like rb_str_resize()

#4 - 06/26/2012 11:47 AM - kosaki (Motohiro KOSAKI)

Can you please mesure performance impact? Because GVL acquire is costly operation. The performance benefit is not obvious to me.

#5 - 06/27/2012 03:23 AM - drbrain (Eric Hodel)

Here is a demonstration video:

http://www.youtube.com/watch?v=uvidgwKyPh4

To skip past the code, use:

http://www.youtube.com/watch?v=uvidgwKyPh4#t=41

#6 - 06/27/2012 06:02 AM - drbrain (Eric Hodel)

=begin
Here is a second benchmark using a gzipped file of 1GB from /dev/random, but only for a single thread using

ruby 2.0.0dev (2012-06-26 trunk 36225) [x86_64-darwin11.4.0]

Code:

```
require 'zlib'
```

```
require 'benchmark'

gz = File.read '1G.gz'

z = Zlib::Inflate.new Zlib::MAX_WBITS + 32 # automatic gzip detection

times = Benchmark.measure do
z.inflate gz
z.finish
end

puts times
```

Without patch:

```
$ ruby20 test.rb
1.850000   0.950000   2.800000 (  2.802966)
[ 13:55 drbrain@YPCMC10014:~/Work/svn/ruby/trunk ]
$ ruby20 test.rb
1.840000   0.950000   2.790000 (  2.791687)
[ 13:55 drbrain@YPCMC10014:~/Work/svn/ruby/trunk ]
$ ruby20 test.rb
1.870000   0.990000   2.860000 (  2.850076)
```

With patch 3:

```
$ make runruby
./miniruby -I./lib -I. -I.ext/common  ./tool/runruby.rb --extout=.ext  -- --disable-gems ./test.rb
1.850000   0.960000   2.810000 (  2.807008)
$ make runruby
./miniruby -I./lib -I. -I.ext/common  ./tool/runruby.rb --extout=.ext  -- --disable-gems ./test.rb
1.840000   0.930000   2.770000 (  2.771213)
$ make runruby
./miniruby -I./lib -I. -I.ext/common  ./tool/runruby.rb --extout=.ext  -- --disable-gems ./test.rb
1.840000   0.970000   2.810000 (  2.817809)
```

So for large files there is no perceptible difference for a single thread.

=end

#7 - 06/27/2012 06:23 AM - drbrain (Eric Hodel)

=begin
This benchmark inflates ~43,000 100KB deflate strings (~1GB total input) using the same ruby version as above.

Code:

```
require 'zlib'
require 'benchmark'

r = Random.new 0

file_count = 2**30 / 100_000 # 100KB chunks in 1GB

deflated = (0..file_count).map do
input = r.bytes 100_000
Zlib::Deflate.deflate input
end

times = Benchmark.measure do
deflated.each do |input|
Zlib::Inflate.inflate input
end
end

puts times
```

Without patch:

```
$ ruby20 test.rb
1.350000   0.040000   1.390000 (  1.378062)
$ ruby20 test.rb
1.330000   0.010000   1.340000 (  1.343311)
$ ruby20 test.rb
1.350000   0.020000   1.370000 (  1.363618)
$ ruby20 test.rb
1.430000   0.030000   1.460000 (  1.464801)
```

With patch:

```
$ make runruby
1.170000   0.020000   1.190000 (  1.198120)
$ make runruby
1.250000   0.020000   1.270000 (  1.273507)
$ make runruby
1.320000   0.010000   1.330000 (  1.333873)
```

So there is a slight increase in performance, probably due to use of realloc vs REALLOC_N
=end

#8 - 06/27/2012 06:47 AM - drbrain (Eric Hodel)

=begin

This benchmarks inflates 100,000 1000byte deflate strings. 1000 bytes is smaller than the default Zlib buffer size, so use of realloc vs REALLOC_N should not matter.

Code:

```
require 'zlib'
require 'benchmark'

r = Random.new 0

file_count = 100_000

deflated = (0..file_count).map do
input = r.bytes 1000
Zlib::Deflate.deflate input
end

times = Benchmark.measure do
deflated.each do |input|
Zlib::Inflate.inflate input
end
end

puts times
```

Without patch:

```
$ for f in jot 5; do ruby20 test.rb; done
0.810000   0.060000   0.870000 (  0.864994)
0.800000   0.050000   0.850000 (  0.863737)
0.800000   0.060000   0.860000 (  0.851056)
0.830000   0.060000   0.890000 (  0.887332)
0.810000   0.060000   0.870000 (  0.866989)
```

With patch:

```
$ for f in jot 5; do make runruby; done
0.780000   0.060000   0.840000 (  0.851944)
0.800000   0.060000   0.860000 (  0.865989)
0.770000   0.050000   0.820000 (  0.823333)
0.770000   0.060000   0.830000 (  0.828246)
0.760000   0.060000   0.820000 (  0.828063)
```

Slight increase in performance.

=end

#9 - 06/27/2012 06:57 AM - drbrain (Eric Hodel)

=begin
Since the last benchmark is so close, this benchmark uses the same deflate file set, but adds GVL contention by deflating the set across four threads. Since no buffer expansion occurs this will benchmark cost of GVL release vs rb_thread_schedule() in present Zlib code.

Code:

```
require 'zlib'
require 'benchmark'

r = Random.new 0

file_count = 100_000

deflated = (0..file_count).map do
input = r.bytes 1000
Zlib::Deflate.deflate input
end

times = Benchmark.measure do
(0..3).map do
Thread.new do
deflated.each do |input|
Zlib::Inflate.inflate input
end
end
end.each do |t|
t.join
end
end

puts times
```

Without patch:

```
$ for f in jot 5; do ruby20 test.rb; done
5.420000   5.970000  11.390000 (  8.162893)
5.400000   6.270000  11.670000 (  8.263046)
5.460000   5.920000  11.380000 (  8.133742)
5.410000   6.290000  11.700000 (  8.289913)
5.500000   6.620000  12.120000 (  8.478085)
```

With patch:

```
$ for f in jot 5; do make runruby; done
5.120000   6.240000  11.360000 (  8.039715)
5.240000   6.260000  11.500000 (  8.097961)
5.280000   5.940000  11.220000 (  8.004246)
5.210000   6.360000  11.570000 (  8.171124)
5.240000   6.200000  11.440000 (  8.054929)
```

Again, slight improvement, but not as great as the video.

The inflate function is able to operate in parallel a small fraction of the time which is able to make up for the cost of GVL release/acquire.
=end

#10 - 06/27/2012 07:29 AM - drbrain (Eric Hodel)

=begin
Re-running this last benchmark with only 10,000 files across 50 runs, using ministat from FreeBSD to calculate the decrease in time, the real run time is reduced 2.6% +/- 1.9% at 99% confidence:

$ ministat -c 99 -s without with
x without

- with +----------+ | + + + x * xx x | | + x + + + + x +xx*xx x * x + x | |+ ++++ x + x*+ * + ++x+x++x*++x* ***+xxx * +x**** * + x x xx| | |_____A_____/ / | /_____A_____| | +----------+ N Min Max Median Avg Stddev x 50 0.61660767 0.71641994 0.66669798 0.66647618 0.022548872
- 50 0.59242272 0.69294882 0.64941692 0.64917859 0.02509235 Difference at 99.0% confidence -0.0172976 +/- 0.0125332 -2.59538% +/- 1.88051% (Student's t, pooled s = 0.0238545)

See http://www.freebsd.org/cgi/man.cgi?query=ministat for the ministat man page

=end

#11 - 06/28/2012 12:58 PM - kosaki (Motohiro KOSAKI)

Thank you. I'm convinced this is good feature. :)
So, +1 from me.

#12 - 07/03/2012 06:03 AM - drbrain (Eric Hodel)

- Status changed from Open to Closed

- % Done changed from 0 to 100

This issue was solved with changeset r36270.
Eric, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.

- ext/zlib/zlib.c (zstream_run): Process zlib streams without GVL. [Feature #6615]
- NEWS: ditto.

Files

zlib.release_gvl.patch	2.32 KB	06/21/2012	drbrain (Eric Hodel)
zlib.release_gvl.2.patch	5.3 KB	06/21/2012	drbrain (Eric Hodel)
zlib.release_gvl.3.patch	5.35 KB	06/26/2012	drbrain (Eric Hodel)