Ruby - Feature #13552

[PATCH 0/2] reimplement ConditionVariable, Queue, SizedQueue using ccan/list

05/10/2017 12:16 AM - normalperson (Eric Wong)

Status:	Closed
Priority:	Normal
Assignee:	normalperson (Eric Wong)
Target version:	2.5

Description

```
The performance improvement increases as the number of waiters
increases, due to avoiding the O(n) behavior of rb_ary_delete on
the waiting thread.  Uncontended queues and condition variables
performance is not altered significantly.

Function entry cost is slightly increased for ConditionVariable,
since the data pointer is separately allocated and not embedded
into the RVALUE slot.

name                   |trunk  |built
-----------------------|------:|------:
vm_thread_condvar1     |  0.858|  0.858
vm_thread_condvar2     |  1.003|  0.804
vm_thread_queue        |  0.131|  0.129
vm_thread_sized_queue  |  0.265|  0.251
vm_thread_sized_queue2|  0.892|  0.859
vm_thread_sized_queue3|  0.879|  0.845
vm_thread_sized_queue4|  0.599|  0.486

Speedup ratio: compare with the result of `trunk' (greater is better)

name                   |built
-----------------------|------:
vm_thread_condvar1     |  0.999
vm_thread_condvar2     |  1.246
vm_thread_queue        |  1.020
vm_thread_sized_queue  |  1.057
vm_thread_sized_queue2|  1.039
vm_thread_sized_queue3|  1.041
vm_thread_sized_queue4|  1.233
```

Associated revisions

Revision ea1ce47fd7f2bc9023e9a1391dbadcfaf9e892ce - 05/19/2017 06:53 PM - Eric Wong

thread_sync.c: rewrite the rest using using ccan/list

The performance improvement increases as the number of waiters
increases, due to avoiding the O(n) behavior of rb_ary_delete on
the waiting thread. Uncontended queues and condition variables
performance is not altered significantly.

Function entry cost is slightly increased for ConditionVariable,
since the data pointer is separately allocated and not embedded
into the RVALUE slot.

[ruby-core:81235] [Feature #13552]

name	trunk	built
vm_thread_condvar1	0.858	0.858
vm_thread_condvar2	1.003	0.804
vm_thread_queue	0.131	0.129
vm_thread_sized_queue	0.265	0.251
vm_thread_sized_queue2	0.892	0.859

name	trunk	built
vm_thread_sized_queue3	0.879	0.845
vm_thread_sized_queue4	0.599	0.486

Speedup ratio: compare with the result of `trunk' (greater is better)

name	built
vm_thread_condvar1	0.999
vm_thread_condvar2	1.246
vm_thread_queue	1.020
vm_thread_sized_queue	1.057
vm_thread_sized_queue2	1.039
vm_thread_sized_queue3	1.041
vm_thread_sized_queue4	1.233

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@58805 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision ea1ce47f - 05/19/2017 06:53 PM - Eric Wong

thread_sync.c: rewrite the rest using using ccan/list

The performance improvement increases as the number of waiters
increases, due to avoiding the O(n) behavior of rb_ary_delete on
the waiting thread. Uncontended queues and condition variables
performance is not altered significantly.

Function entry cost is slightly increased for ConditionVariable,
since the data pointer is separately allocated and not embedded
into the RVALUE slot.

[ruby-core:81235] [Feature #13552]

name	trunk	built
vm_thread_condvar1	0.858	0.858
vm_thread_condvar2	1.003	0.804
vm_thread_queue	0.131	0.129
vm_thread_sized_queue	0.265	0.251
vm_thread_sized_queue2	0.892	0.859
vm_thread_sized_queue3	0.879	0.845
vm_thread_sized_queue4	0.599	0.486

Speedup ratio: compare with the result of `trunk' (greater is better)

name	built
vm_thread_condvar1	0.999
vm_thread_condvar2	1.246
vm_thread_queue	1.020
vm_thread_sized_queue	1.057
vm_thread_sized_queue2	1.039
vm_thread_sized_queue3	1.041
vm_thread_sized_queue4	1.233

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@58805 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 17bf0c0001179858aacabbfe1b8bf2960e6b9083 - 07/21/2017 07:06 PM - Eric Wong

NEWS: add entries for thread_sync.c changes

I'm slightly worried about some external code subclassing
ConditionVariable, Queue, and SizedQueue and relying on them
being Structs. However, they only started being Structs with
Ruby 2.1, and were implemented in pure Ruby before that; so
hopefully nobody notices that implementation detail.

Also, note the Mutex change as it may affect program design
when space can be saved.

- NEWS: entries for [Feature #13552] and [Feature #13517]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59385 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 17bf0c00 - 07/21/2017 07:06 PM - Eric Wong

NEWS: add entries for thread_sync.c changes

I'm slightly worried about some external code subclassing
ConditionVariable, Queue, and SizedQueue and relying on them
being Structs. However, they only started being Structs with
Ruby 2.1, and were implemented in pure Ruby before that; so
hopefully nobody notices that implementation detail.

Also, note the Mutex change as it may affect program design
when space can be saved.

- NEWS: entries for [Feature #13552] and [Feature #13517]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@59385 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

History

#1 - 05/10/2017 12:17 AM - normalperson (Eric Wong)

- File 0002-thread_sync.c-rewrite-the-rest-using-using-ccan-list.patch added

#2 - 05/10/2017 12:19 AM - normalperson (Eric Wong)

pull request:

The following changes since commit 6ad7c53ba9fb688ea1070a2319a64f0cc32c08e8:

test/thread: relax internal implementation check in error message (2017-05-09 19:52:10 +0000)

are available in the git repository at:

git://80x24.org/ruby.git sync-list

for you to fetch changes up to 4d77449e1c832d4398cdc07ef10b57e55bea1b81:

thread_sync.c: rewrite the rest using using ccan/list (2017-05-09 20:42:50 +0000)

Eric Wong (2):
thread_sync.c: rename mutex_waiter struct to sync_waiter
thread_sync.c: rewrite the rest using using ccan/list

thread_sync.c | 487 ++--------------------
1 file changed, 324 insertions(+), 163 deletions(-)

#3 - 05/18/2017 05:21 PM - normalperson (Eric Wong)

normalperson@yhbt.net wrote:

```
thread_sync.c: rename mutex_waiter struct to sync_waiter
thread_sync.c: rewrite the rest using using ccan/list
```

Any comment? Rebased patches against current trunk (r58783) available here:

```
https://80x24.org/spew/20170516033841.1795-1-e@80x24.org/raw
https://80x24.org/spew/20170516033841.1795-2-e@80x24.org/raw
```

Thanks.

> Feature #13552: [PATCH 0/2] reimplement ConditionVariable, Queue, SizedQueue using ccan/list
> https://bugs.ruby-lang.org/issues/13552#change-64734

#4 - 05/19/2017 02:24 AM - ko1 (Koichi Sasada)

- Status changed from Open to Assigned

- Assignee set to normalperson (Eric Wong)

- Target version set to 2.5

Sorry for late response.

Only one comment (maybe you passes all of tests, right?)

New data type should be RUBY_TYPED_WB_PROTECTED (they need to use write barriers correctly).
Do you want to try or should I modify?

Thanks,
Koichi

#5 - 05/19/2017 03:51 AM - normalperson (Eric Wong)

ko1@atdot.net wrote:

> Sorry for late response.

No problem.

> Only one comment (maybe you passes all of tests, right?)

Of course :)

> New data type should be RUBY_TYPED_WB_PROTECTED (they need to use write barriers correctly).
> Do you want to try or should I modify?

I'm still not very familiar with RGenGC, but here is my try:

```
https://80x24.org/spew/20170519034419.GA29820@whir/raw
```

I'm not sure how this helps performance, however. The Arrays
are constantly changing with push/pop and RGenGC works best for
stable (unchanging) objects (correct?)

Also, does setting RUBY_TYPED_WB_PROTECTED make sense for
rb_condvar and rb_mutex_t? They store no Ruby objects and
have no dmark callback.

Thanks.

#6 - 05/19/2017 06:39 AM - ko1 (Koichi Sasada)

> https://80x24.org/spew/20170519034419.GA29820@whir/raw

Thank you. Adding const helps us to recognize.

```
PACKED_STRUCT_UNALIGNED(struct rb_queue {
    struct list_head waitq;
    const VALUE que;
    int num_waiting;
});
```

> I'm not sure how this helps performance, however. The Arrays
> are constantly changing with push/pop and RGenGC works best for
> stable (unchanging) objects (correct?)

Sorry, I can't understand your question.

Could you give me your question in other words?

> Also, does setting RUBY_TYPED_WB_PROTECTED make sense for
> rb_condvar and rb_mutex_t? They store no Ruby objects and
> have no dmark callback.

Yes, please. not wb protected objects become roots for all of minor gc.
No write is the best wb protected object.

#7 - 05/19/2017 08:11 AM - normalperson (Eric Wong)

ko1@atdot.net wrote:

> > https://80x24.org/spew/20170519034419.GA29820@whir/raw
>
> Thank you. Adding const helps us to recognize.
>
> ```
> PACKED_STRUCT_UNALIGNED(struct rb_queue {
>     struct list_head waitq;
>     const VALUE que;
>     int num_waiting;
> });
> ```

Thank you for that advice! I will update tomorrow.

> ```
> > I'm not sure how this helps performance, however. The Arrays
> > are constantly changing with push/pop and RGenGC works best for
> > stable (unchanging) objects (correct?)
>
> Sorry, I can't understand your question.
> Could you give me your question in other words?
> ```

Generational GC tries to avoid marking since "old" generation
does not change references.

However, the ->que in Queue/SizedQueue is always changing
because threads push/pop. When references are always changing
in Queues, so GC needs mark ->que frequently.

> > Also, does setting RUBY_TYPED_WB_PROTECTED make sense for
> > rb_condvar and rb_mutex_t? They store no Ruby objects and
> > have no dmark callback.
>
> Yes, please. not wb protected objects become roots for all of minor gc.
> No write is the best wb protected object.

Good to know! I will update and commit tomorrow.

#8 - 05/19/2017 06:53 PM - Anonymous

- Status changed from Assigned to Closed

Applied in changeset trunk|r58805.

---

thread_sync.c: rewrite the rest using using ccan/list

The performance improvement increases as the number of waiters
increases, due to avoiding the O(n) behavior of rb_ary_delete on
the waiting thread. Uncontended queues and condition variables
performance is not altered significantly.

Function entry cost is slightly increased for ConditionVariable,
since the data pointer is separately allocated and not embedded
into the RVALUE slot.

[ruby-core:81235] [Feature #13552]

name	trunk	built
vm_thread_condvar1	0.858	0.858
vm_thread_condvar2	1.003	0.804
vm_thread_queue	0.131	0.129
vm_thread_sized_queue	0.265	0.251
vm_thread_sized_queue2	0.892	0.859
vm_thread_sized_queue3	0.879	0.845
vm_thread_sized_queue4	0.599	0.486

Speedup ratio: compare with the result of `trunk' (greater is better)

name	built
vm_thread_condvar1	0.999
vm_thread_condvar2	1.246
vm_thread_queue	1.020
vm_thread_sized_queue	1.057
vm_thread_sized_queue2	1.039
vm_thread_sized_queue3	1.041
vm_thread_sized_queue4	1.233

Files

0001-thread_sync.c-rename-mutex_waiter-struct-to-sync_wai.patch	1.63 KB	05/10/2017	normalperson (Eric Wong)
0002-thread_sync.c-rewrite-the-rest-using-using-ccan-list.patch	21.3 KB	05/10/2017	normalperson (Eric Wong)