Ruby master - Feature #16984

Remove write barrier exemption for T_ICLASS

06/25/2020 01:03 AM - alanwu (Alan Wu)

Status:	Closed
Priority:	Normal
Assignee:	
Target version:	

Description

Currently, iclasses are "shady", or not protected by write barriers. Because of that, the GC needs to spend more time marking these objects than otherwise. Let's insert write barriers for iclasses.

Applications that use modules heavily should see reduction in GC time as they have a significant number of live iclasses on the heap.

- Put logic for iclass method table ownership into a function
- Remove calls to WB_UNPROTECT and insert write barriers for iclasses

The change relies on the following invariant: for any non origin iclass I, RCLASS_M_TBL(I) == RCLASS_M_TBL(RBasic(I)->klass). This invariant did not hold prior to 98286e9 for classes and modules that have prepended modules.

Patch: https://github.com/ruby/ruby/pull/3410

This is the second version of this change. It's much simpler and it doesn't introduce new garbage collected objects. I realized that despite saving a pointer to some other object's method table, iclasses don't mark the method tables. So, for each method table, there is an unique object that's responsible for marking it. Since write barriers are only needed for the object that is marking the newly written value (correct me if I'm wrong here), having an unique object that marks the tables makes things straight forward.

The numbers from v1 of this patch was a bit inflated because we were allocating an excessive amount of iclasses.
I measured again and an app that has an approximately 250MiB heap saw a 22% reduction in minor GC time.

Credits to tenderlovemaking (Aaron Patterson) for motivating this change.

Associated revisions

Revision 264e4cd0 - 08/17/2020 09:17 PM - alanwu (Alan Wu)

Remove write barrier exemption for T_ICLASS

Before this commit, iclasses were "shady", or not protected by write barriers. Because of that, the GC needs to spend more time marking these objects than otherwise.

Applications that make heavy use of modules should see reduction in GC time as they have a significant number of live iclasses on the heap.

- Put logic for iclass method table ownership into a function
- Remove calls to WB_UNPROTECT and insert write barriers for iclasses

This commit relies on the following invariant: for any non oirigin iclass I, RCLASS_M_TBL(I) == RCLASS_M_TBL(RBasic(I)->klass). This invariant did not hold prior to 98286e9 for classes and modules that have prepended modules.

[Feature #16984]

History

#1 - 06/27/2020 07:12 PM - ko1 (Koichi Sasada)

Thank you for great work. This kind of hack can cause BUGs easily.

> Private app A's heap size is about 22 MiB compared to B's 250 MiB.

Could you measure the memory/objects consumption before and after this patch if it is not difficult?
Maybe no problem, but I want to confirm.

#2 - 06/30/2020 07:13 AM - alanwu (Alan Wu)

> Could you measure the memory/objects consumption before and after this patch if it is not difficult?

I took measurements on app B. It's a large Rails app with lots of classes and modules.
The amount of retained memory is not deterministic unfortunately, so I can only give a rough summary.

	Change to median	Change to average
GC::Profiler "Total Size"	7 MiB	1 MiB
VmRSS from the /proc	4 MiB	-5 MiB

The increase to GC heap size makes sense, I expect about 5 MiB more objects given the number of classes and modules in the app. There is not much change to RSS I guess because the patch moves what used to be on the malloc heap to the GC heap.

#3 - 07/13/2020 09:51 PM - alanwu (Alan Wu)

- Description updated

Edit: I noticed that T_ICLASS wasn't marking the shared method and constant table on master. My notes about reducing the number of gc_mark references on the heap were incorrect.

#4 - 08/12/2020 06:28 AM - ko1 (Koichi Sasada)

sorry I didn't check it.

Thank you, ~10 MB in "B's 250 MiB." is not problem I think.
Could you merge it if you don't have any trouble more?

#5 - 08/13/2020 03:31 AM - alanwu (Alan Wu)

- Description updated

- Subject changed from Remove write barrier examption for T_ICLASS to Remove write barrier exemption for T_ICLASS

#6 - 08/13/2020 03:51 AM - alanwu (Alan Wu)

- Description updated

#7 - 08/13/2020 03:55 AM - alanwu (Alan Wu)

- Description updated

#8 - 08/17/2020 09:16 PM - alanwu (Alan Wu)

- Description updated

#9 - 08/17/2020 09:18 PM - alanwu (Alan Wu)

- Status changed from Open to Closed

Applied in changeset git|264e4cd04fbcdcb739a1ff9a84e19afe66005cb2.

Remove write barrier exemption for T_ICLASS

Before this commit, iclasses were "shady", or not protected by write barriers. Because of that, the GC needs to spend more time marking these

objects than otherwise.

Applications that make heavy use of modules should see reduction in GC time as they have a significant number of live iclasses on the heap.

- Put logic for iclass method table ownership into a function
- Remove calls to WB_UNPROTECT and insert write barriers for iclasses

This commit relies on the following invariant: for any non oirigin iclass I, RCLASS_M_TBL(I) == RCLASS_M_TBL(RBasic(I)->klass). This invariant did not hold prior to 98286e9 for classes and modules that have prepended modules.

[Feature #16984]