Ruby trunk - Feature #6309

Add a reference queue for weak references

04/17/2012 05:10 PM - headius (Charles Nutter)

Status:	Assigned	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:		

Description

Most interesting uses of WeakRef are much harder to do efficiently without a reference queue.

A reference queue, as implemented by the JVM, is basically a queue into which weak references are placed some time after the object they refer to has been collected. The queue can be polled cheaply to look for collected references.

A simple example of usage can be seen in the weakling gem, with an efficient implementation of an ID hash:
https://github.com/headius/weakling/blob/master/lib/weakling/collections.rb

Notice the _cleanup method is called for every operation, to keep the hash clear of dead references. Failure to have a _cleanup method would mean the hash grows without bounds.

_cleanup cannot be implemented efficiently on MRI at present because there's no reference queue implementation. On MRI, _cleanup would have to perform a linear scan of all stored values periodically to search for dead references. For a heavily used hash with many live values, this becomes a very expensive operation.

It's probably possible to implement reference queues efficiently atop the new ObjectSpace::WeakMap internals, since it already keeps track of weak references and can run code when a weak reference no longer refers to a live object.

Related issues:

Related to CommonRuby - Feature #6308: Eliminate delegation from WeakRef	**Assigned**	**04/17/2012**

History

#1 - 04/20/2012 04:51 AM - mame (Yusuke Endoh)

- Status changed from Open to Feedback

There is no maintainer for weakref, so please create a patch yourself!
We may import it if matz accepts.

BTW, the name "reference queue" is very bad, I think.

--
Yusuke Endoh mame@tsg.ne.jp

#2 - 04/28/2012 02:47 PM - headius (Charles Nutter)

Ok, fair enough.

Here is a *very primitive* modification of the current weakref.rb to support a reference queue. I need to stress that I don't think this is the best way to implement it; a hook into the GC cycle that inserts weakrefs into a purpose-built reference queue would be better than using finalizers in this way. But the API would largely work the same.

Patch: https://gist.github.com/2516338

Example usage: https://gist.github.com/2516355

This works mostly like I expect a reference queue to work, but there are many inefficiencies here:

- Polling the reference queue needs to be as close to free as possible. The current Queue implementation raises an exception when empty, which is very far from being free.
- A Ruby-level finalizer is much more expensive than a purpose-built native GC hook would be.
- A Ruby-based Queue is much more expensive than a purpose-built reference queue would be.

I know that in past discussions about improving weakref support in Ruby there were C-level patches to add all the features I'm looking for, and I'll try to dig up those discussions and patches. But hopefully this illustrates what I'm looking for in a primitive way.

#3 - 05/03/2012 11:29 AM - mame (Yusuke Endoh)

Ah, I knew what you are proposing by seeing Javadoc:

http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/ref/ReferenceQueue.html
http://docs.oracle.com/javase/6/docs/api/java/lang/ref/WeakReference.html#WeakReference(T, java.lang.ref.ReferenceQueue)

I don't know the (real-world) use case of the feature, though.

Anyway, I mean I'd like you to create a patch written *in C*.
If there is a patch that we can review and import "as is",
I will be happy to assign this ticket to some core committers,
such as ko1 and kosaki.

--
Yusuke Endoh mame@tsg.ne.jp

#4 - 05/03/2012 11:50 AM - mame (Yusuke Endoh)

- Status changed from Feedback to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

On second thought, the proposal should first get an approval from matz. Sorry. Assigning this to him.
Still, it would be helpful to show a concrete use case, I think.

--
Yusuke Endoh mame@tsg.ne.jp

#5 - 05/04/2012 05:59 AM - headius (Charles Nutter)

I linked to a concrete use case in the original report...an implementation of a "weak ID map" entirely in Ruby without scanning for dead references:
https://github.com/headius/weakling/blob/master/lib/weakling/collections.rb

It is not possible to implement weak data structures efficiently without a reference queue, since you would be forced to periodically do an O(N) scan for dead references to clean them out.

#6 - 11/17/2012 01:26 AM - headius (Charles Nutter)

Seven months and no activity. Can we get a reference queue in Ruby 2.0 please? I believe it could be added to weakref.rb using 2.0's WeakHash, or built atop the C code that implements WeakHash (since it contains most of a reference queue implementation already).

#7 - 11/24/2012 10:50 AM - mame (Yusuke Endoh)

- Target version set to 2.6

#8 - 03/16/2013 02:27 AM - headius (Charles Nutter)

I request a ruling by matz about adding a ReferenceQueue to weakref.rb.

#9 - 09/27/2013 08:19 PM - headius (Charles Nutter)

I again request approval from matz to add this feature :-) Can we do it for 2.1, please?

#10 - 09/30/2013 10:21 PM - headius (Charles Nutter)

- Target version changed from 2.6 to 2.1.0

#11 - 10/01/2013 10:23 AM - headius (Charles Nutter)

Put my patch plus a test in a PR: https://github.com/ruby/ruby/pull/408

Unfortunately the test doesn't pass, and I think it should. I'm confused as to why it fails.

#12 - 12/14/2013 01:33 PM - nobu (Nobuyoshi Nakada)

My PR: https://github.com/ruby/ruby/pull/480

#13 - 12/30/2013 01:07 AM - headius (Charles Nutter)

Sorry this didn't get into 2.1 and I was unable to review. December was probably too late to get it in anyway.

Nobu's patch looks fine. If other ruby-core folks really want to keep Weakref as-is for compatibility and introduce a new type, I guess that's the way we'll have to go.

A couple comments.

- ext/weakref should be deprecated and warn when loaded once ext/weak is in place.
- Is there interest in other possible reference types? If so, having a namespace for these new references would be less cumbersome. I will describe the reference types on JVM below.

On JVM, there is WeakReference, of course. There's also two others that are useful:

- SoftReference is a reference cleared less frequently than a weak reference. This is JVM implememtation-specific, but on OpenJDK it is a combination of heap pressure (if the heap has to be expanded, soft references are cleared) or if the soft reference is not traversed for some period of time (configurable as some number of ms/MB of heap).
- PhantomReference is similar to weak reference in life cycle, but is not traversible and only useful when combined with a queue. This is lighter-weight than weak reference, since it does not have to be cleared when the object is collected; it just needs to be enqueued. It also does not impact GC cycles as much, since it does not count as a strong or weak traversible reference.

If we might want these types of references in the future, it may be good to have a Reference namespace, a la Reference::Weak, Reference::Soft, etc.

#14 - 12/30/2013 01:07 AM - headius (Charles Nutter)

- Target version changed from 2.1.0 to 2.2.0

#15 - 01/05/2018 09:00 PM - naruse (Yui NARUSE)

- Target version deleted (2.2.0)