Ruby master - Feature #15413

unmarkable C stack (3rd stack)

12/14/2018 09:32 PM - normalperson (Eric Wong)

Status:	Open	
Priority:	Normal	
Assignee:		
Target version:		

Description

The current machine (C) stack can get pretty big for some C functions
(rb_ensure, rb_f_select/rb_thread_fd_select/...). This is harmful when we stop
a fiber/thread and all that stack becomes eligible for marking.

We should experiment a bump allocator for temporary allocations which
behaves like the stack, but does not get marked by GC. VALUEs will continue
to be allocated on normal C stack, but non-VALUE stuff can go to the
unmarkable machine stack.

Maybe we call it "UMMS" for Un-Markable Machine Stack

We cannot remove marking of the current C stack for compatibility;
but we can transition existing C code to use UMMS.

I probably won't be around to work on it for 2.7, unfortunately.

History

#1 - 12/15/2018 01:04 AM - ko1 (Koichi Sasada)

On 2018/12/15 6:32, normalperson@yhbt.net wrote:

> We should experiment a bump allocator for temporary allocations which
> behaves like the stack, but does not get marked by GC. VALUEs will continue
> to be allocated on normal C stack, but non-VALUE stuff can go to the
> unmarkable machine stack.

memory space for alloca()?

--
// SASADA Koichi at atdot dot net

#2 - 12/15/2018 01:13 AM - normalperson (Eric Wong)

Koichi Sasada ko1@atdot.net wrote:

> On 2018/12/15 6:32, normalperson@yhbt.net wrote:
>
> > We should experiment a bump allocator for temporary allocations which
> > behaves like the stack, but does not get marked by GC. VALUEs will continue
> > to be allocated on normal C stack, but non-VALUE stuff can go to the
> > unmarkable machine stack.
>
> memory space for alloca()?

Partially, but non-alloca structs also add up, too. The
select()-based auto-fiber had trouble because of select_args and
rb_ensure_list_t sizes, so I needed to move some allocations around.

rb_ensure_list_t overhead is only for callcc, anyways, so maybe
we can also have a lightweight version of rb_ensure when we
don't have user code which may call callcc. We use
rb_ensure for sleeping in queue/mutex/waitpid/... and other
places which never switch stack with callcc.