Ruby master - Feature #10341

Fiber switch performance improvements

10/08/2014 12:16 PM - nome (Knut Franke)

Status:	Closed	
Priority:	Normal	
Assignee:		
Target version:		

Description

The attached patches increase performance of switches between Fibers by ~17% on my test system (Linux, gcc 4.8.2).

Patches 1-3 are purely cosmetic, but included here because submitting them separately would cause conflicts between the patch sets. If these are rejected, I can prepare stand-alone versions of 4/5.

Patch 4 yields the most significant performance increase (~12%). The benefit of patch 5 is lower, and it's a larger change; so this one could optionally be omitted.

```
require 'benchmark'

fib = Fiber.new do
    loop { Fiber.yield }
end

Benchmark.bm do |bm|
    3.times do
        results << bm.report { 10_000_000.times { fib.resume } }
    end
    avg = results.inject(:+) / results.size
        [avg]
end
```

Raw benchmarking results:

```
trunk@47827 - 7.59s
patch 4           - 6.59s (87% of trunk)
patch 4+5      - 6.33s (83% of trunk)
```

Associated revisions

Revision 6f6238a7 - 10/15/2014 10:34 PM - normal

cont.c: Remove unused prev_fiber/next_fiber fields

They were added in r19890 (8a0d53a), with the explanation that it's a
double linked list of fibers in the same thread, but without any code
using them.

- cont.c (rb_fiber_t): remove prev_fiber/next_fiber (fiber_link_join, fiber_link_remove): remove functions (fiber_free, fiber_init, root_fiber_alloc): remove references to removed fields and functions [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47959 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 47959 - 10/15/2014 10:34 PM - normal

cont.c: Remove unused prev_fiber/next_fiber fields

They were added in r19890 (8a0d53a), with the explanation that it's a
double linked list of fibers in the same thread, but without any code
using them.

- cont.c (rb_fiber_t): remove prev_fiber/next_fiber (fiber_link_join, fiber_link_remove): remove functions (fiber_free, fiber_init, root_fiber_alloc): remove references to removed fields and functions [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47959 - 10/15/2014 10:34 PM - normalperson (Eric Wong)

cont.c: Remove unused prev_fiber/next_fiber fields

They were added in r19890 (8a0d53a), with the explanation that it's a
double linked list of fibers in the same thread, but without any code
using them.

- cont.c (rb_fiber_t): remove prev_fiber/next_fiber (fiber_link_join, fiber_link_remove): remove functions (fiber_free, fiber_init, root_fiber_alloc): remove references to removed fields and functions [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47959 - 10/15/2014 10:34 PM - normal

cont.c: Remove unused prev_fiber/next_fiber fields

They were added in r19890 (8a0d53a), with the explanation that it's a
double linked list of fibers in the same thread, but without any code
using them.

- cont.c (rb_fiber_t): remove prev_fiber/next_fiber (fiber_link_join, fiber_link_remove): remove functions (fiber_free, fiber_init, root_fiber_alloc): remove references to removed fields and functions [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47959 - 10/15/2014 10:34 PM - normal

cont.c: Remove unused prev_fiber/next_fiber fields

They were added in r19890 (8a0d53a), with the explanation that it's a
double linked list of fibers in the same thread, but without any code
using them.

- cont.c (rb_fiber_t): remove prev_fiber/next_fiber (fiber_link_join, fiber_link_remove): remove functions (fiber_free, fiber_init, root_fiber_alloc): remove references to removed fields and functions [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47959 - 10/15/2014 10:34 PM - normal

cont.c: Remove unused prev_fiber/next_fiber fields

They were added in r19890 (8a0d53a), with the explanation that it's a
double linked list of fibers in the same thread, but without any code
using them.

- cont.c (rb_fiber_t): remove prev_fiber/next_fiber (fiber_link_join, fiber_link_remove): remove functions (fiber_free, fiber_init, root_fiber_alloc): remove references to removed fields and functions [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47959 - 10/15/2014 10:34 PM - normal

cont.c: Remove unused prev_fiber/next_fiber fields

They were added in r19890 (8a0d53a), with the explanation that it's a
double linked list of fibers in the same thread, but without any code
using them.

- cont.c (rb_fiber_t): remove prev_fiber/next_fiber (fiber_link_join, fiber_link_remove): remove functions (fiber_free, fiber_init, root_fiber_alloc): remove references to removed fields and functions [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 0bd492c6 - 10/15/2014 10:34 PM - normal

cont.c: Code cleanup in fiber_switch/fiber_store

Defragment code blocks depending on FIBER_USE_NATIVE in order to make
the control flow (which is already non-trivial due to nonlocal jumps) in
each case more comprehensible.

Remove some unreachable code from fiber_switch (we've already excluded
the case (th->fiber == fibval) at the start of the function).

Remove call to rb_fiber_current which happened a few lines after
accessing GET_THREAD()->fiber directly (so if that's ever 0 we're
already screwed).

Author: Knut Franke Knut.Franke@gmx.de

- cont.c (fiber_store, fiber_switch): simplify [ruby-core:65518] [Feature #10341]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47961 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 47961 - 10/15/2014 10:34 PM - normal

cont.c: Code cleanup in fiber_switch/fiber_store

Defragment code blocks depending on FIBER_USE_NATIVE in order to make
the control flow (which is already non-trivial due to nonlocal jumps) in
each case more comprehensible.

Remove some unreachable code from fiber_switch (we've already excluded
the case (th->fiber == fibval) at the start of the function).

Remove call to rb_fiber_current which happened a few lines after
accessing GET_THREAD()->fiber directly (so if that's ever 0 we're
already screwed).

Author: Knut Franke Knut.Franke@gmx.de

- cont.c (fiber_store, fiber_switch): simplify [ruby-core:65518] [Feature #10341]

Revision 47961 - 10/15/2014 10:34 PM - normalperson (Eric Wong)

cont.c: Code cleanup in fiber_switch/fiber_store

Defragment code blocks depending on FIBER_USE_NATIVE in order to make
the control flow (which is already non-trivial due to nonlocal jumps) in
each case more comprehensible.

Remove some unreachable code from fiber_switch (we've already excluded
the case (th->fiber == fibval) at the start of the function).

Remove call to rb_fiber_current which happened a few lines after
accessing GET_THREAD()->fiber directly (so if that's ever 0 we're
already screwed).

Author: Knut Franke Knut.Franke@gmx.de

- cont.c (fiber_store, fiber_switch): simplify [ruby-core:65518] [Feature #10341]

Revision 47961 - 10/15/2014 10:34 PM - normal

cont.c: Code cleanup in fiber_switch/fiber_store

Defragment code blocks depending on FIBER_USE_NATIVE in order to make
the control flow (which is already non-trivial due to nonlocal jumps) in
each case more comprehensible.

Remove some unreachable code from fiber_switch (we've already excluded
the case (th->fiber == fibval) at the start of the function).

Remove call to rb_fiber_current which happened a few lines after
accessing GET_THREAD()->fiber directly (so if that's ever 0 we're
already screwed).

Author: Knut Franke Knut.Franke@gmx.de

- cont.c (fiber_store, fiber_switch): simplify [ruby-core:65518] [Feature #10341]

Revision 47961 - 10/15/2014 10:34 PM - normal

cont.c: Code cleanup in fiber_switch/fiber_store

Defragment code blocks depending on FIBER_USE_NATIVE in order to make
the control flow (which is already non-trivial due to nonlocal jumps) in
each case more comprehensible.

Remove some unreachable code from fiber_switch (we've already excluded
the case (th->fiber == fibval) at the start of the function).

Remove call to rb_fiber_current which happened a few lines after
accessing GET_THREAD()->fiber directly (so if that's ever 0 we're
already screwed).

Author: Knut Franke Knut.Franke@gmx.de

- cont.c (fiber_store, fiber_switch): simplify [ruby-core:65518] [Feature #10341]

Revision 47961 - 10/15/2014 10:34 PM - normal

cont.c: Code cleanup in fiber_switch/fiber_store

Defragment code blocks depending on FIBER_USE_NATIVE in order to make
the control flow (which is already non-trivial due to nonlocal jumps) in
each case more comprehensible.

Remove some unreachable code from fiber_switch (we've already excluded
the case (th->fiber == fibval) at the start of the function).

Remove call to rb_fiber_current which happened a few lines after
accessing GET_THREAD()->fiber directly (so if that's ever 0 we're
already screwed).

Author: Knut Franke Knut.Franke@gmx.de

- cont.c (fiber_store, fiber_switch): simplify [ruby-core:65518] [Feature #10341]

Revision 47961 - 10/15/2014 10:34 PM - normal

cont.c: Code cleanup in fiber_switch/fiber_store

Defragment code blocks depending on FIBER_USE_NATIVE in order to make
the control flow (which is already non-trivial due to nonlocal jumps) in
each case more comprehensible.

Remove some unreachable code from fiber_switch (we've already excluded
the case (th->fiber == fibval) at the start of the function).

Remove call to rb_fiber_current which happened a few lines after
accessing GET_THREAD()->fiber directly (so if that's ever 0 we're
already screwed).

Author: Knut Franke Knut.Franke@gmx.de

- cont.c (fiber_store, fiber_switch): simplify [ruby-core:65518] [Feature #10341]

Revision baeb94fa - 10/15/2014 10:34 PM - normal

cont.c: Small code cleanup

Remove variable that is used only once, several lines after
initialization.

- cont.c (cont_capture): remove unnecessary variable [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47962 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 47962 - 10/15/2014 10:34 PM - normal

cont.c: Small code cleanup

Remove variable that is used only once, several lines after
initialization.

- cont.c (cont_capture): remove unnecessary variable [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47962 - 10/15/2014 10:34 PM - normalperson (Eric Wong)

cont.c: Small code cleanup

Remove variable that is used only once, several lines after
initialization.

- cont.c (cont_capture): remove unnecessary variable [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47962 - 10/15/2014 10:34 PM - normal

cont.c: Small code cleanup

Remove variable that is used only once, several lines after
initialization.

- cont.c (cont_capture): remove unnecessary variable [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47962 - 10/15/2014 10:34 PM - normal

cont.c: Small code cleanup

Remove variable that is used only once, several lines after
initialization.

- cont.c (cont_capture): remove unnecessary variable [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47962 - 10/15/2014 10:34 PM - normal

cont.c: Small code cleanup

Remove variable that is used only once, several lines after
initialization.

- cont.c (cont_capture): remove unnecessary variable [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 47962 - 10/15/2014 10:34 PM - normal

cont.c: Small code cleanup

Remove variable that is used only once, several lines after
initialization.

- cont.c (cont_capture): remove unnecessary variable [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

Revision 71fcbf22 - 10/15/2014 10:35 PM - normal

cont.c (cont_save_thread): Sparse copying of thread data

Instead of copying the complete rb_thread_t struct (almost a kB),
selectively copy only those fields that will be needed later on.

- cont.c (rb_context_t): comment on saved_thread (cont_save_thread): sparse copy (cont_init): copy extra fields (fiber_init): use current thread VM stack size [ruby-core:65518] [Feature #10341]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47963 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 47963 - 10/15/2014 10:35 PM - normal

cont.c (cont_save_thread): Sparse copying of thread data

Instead of copying the complete rb_thread_t struct (almost a kB),
selectively copy only those fields that will be needed later on.

- cont.c (rb_context_t): comment on saved_thread (cont_save_thread): sparse copy (cont_init): copy extra fields (fiber_init): use current thread VM stack size [ruby-core:65518] [Feature #10341]

Revision 47963 - 10/15/2014 10:35 PM - normalperson (Eric Wong)

cont.c (cont_save_thread): Sparse copying of thread data

Instead of copying the complete rb_thread_t struct (almost a kB),
selectively copy only those fields that will be needed later on.

- cont.c (rb_context_t): comment on saved_thread (cont_save_thread): sparse copy (cont_init): copy extra fields (fiber_init): use current thread VM stack size [ruby-core:65518] [Feature #10341]

Revision 47963 - 10/15/2014 10:35 PM - normal

cont.c (cont_save_thread): Sparse copying of thread data

Instead of copying the complete rb_thread_t struct (almost a kB),
selectively copy only those fields that will be needed later on.

- cont.c (rb_context_t): comment on saved_thread (cont_save_thread): sparse copy (cont_init): copy extra fields (fiber_init): use current thread VM stack size [ruby-core:65518] [Feature #10341]

Revision 47963 - 10/15/2014 10:35 PM - normal

cont.c (cont_save_thread): Sparse copying of thread data

Instead of copying the complete rb_thread_t struct (almost a kB),
selectively copy only those fields that will be needed later on.

- cont.c (rb_context_t): comment on saved_thread (cont_save_thread): sparse copy (cont_init): copy extra fields (fiber_init): use current thread VM stack size [ruby-core:65518] [Feature #10341]

Revision 47963 - 10/15/2014 10:35 PM - normal

cont.c (cont_save_thread): Sparse copying of thread data

Instead of copying the complete rb_thread_t struct (almost a kB),
selectively copy only those fields that will be needed later on.

- cont.c (rb_context_t): comment on saved_thread (cont_save_thread): sparse copy (cont_init): copy extra fields (fiber_init): use current thread VM stack size [ruby-core:65518] [Feature #10341]

Revision 47963 - 10/15/2014 10:35 PM - normal

cont.c (cont_save_thread): Sparse copying of thread data

Instead of copying the complete rb_thread_t struct (almost a kB),
selectively copy only those fields that will be needed later on.

- cont.c (rb_context_t): comment on saved_thread (cont_save_thread): sparse copy (cont_init): copy extra fields (fiber_init): use current thread VM stack size [ruby-core:65518] [Feature #10341]

Revision 5c8c88a3 - 10/15/2014 10:35 PM - normal

cont.c: Optimize fiber_switch callees

Remove some unnecessary VALUE/struct conversions and aggressively inline
functions used during fiber_switch. Either of these changes alone does
not yield significant performance increase, but in combination they
improve performance by ~6%.

Arguably, removal of separate VALUE/rb_fiber_t* variables also makes the
code more readable in a few places.

- vm_core.h: declare rb_fiber_t typedef (rb_thread_t): fiber and root_fiber become rb_fiber_t * (from VALUE)
- vm.c (rb_thread_mark): use rb_fiber_mark_self
- cont.c (rb_fiber_t): prev becomes rb_fiber_t * (from VALUE) (cont_mark, cont_free): simplify conditions (rb_fiber_mark_self): new function (fiber_mark): use rb_fiber_mark_self (cont_save_thread, cont_restore_thread): inline (cont_restore_thread): simplify (fiber_setcontext): simplify conditions (rb_cont_call): remove dereference (fiber_t_alloc): update for rb_fiber_t->prev type change (rb_fiber_start): ditto (fiber_current): extract from rb_fiber_current (return_fiber): move, simplify type checks (rb_fiber_current): use fiber_current (fiber_store): simplify type checks (fiber_switch): ditto, simplify call to fiber_setcontext, use fiber_current (rb_fiber_transfer): update for type changes (rb_fiber_terminate): move, use fiber_switch (rb_fiber_resume): update for type changes (rb_fiber_reset_root_local_storage): ditto (rb_fiber_yield): use rb_fiber_switch instead of rb_fiber_transfer (rb_fiber_m_transfer): ditto [ruby-core:65518] [Feature #10341]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47964 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 47964 - 10/15/2014 10:35 PM - normal

cont.c: Optimize fiber_switch callees

Remove some unnecessary VALUE/struct conversions and aggressively inline
functions used during fiber_switch. Either of these changes alone does
not yield significant performance increase, but in combination they
improve performance by ~6%.

Arguably, removal of separate VALUE/rb_fiber_t* variables also makes the
code more readable in a few places.

- vm_core.h: declare rb_fiber_t typedef (rb_thread_t): fiber and root_fiber become rb_fiber_t * (from VALUE)
- vm.c (rb_thread_mark): use rb_fiber_mark_self
- cont.c (rb_fiber_t): prev becomes rb_fiber_t * (from VALUE) (cont_mark, cont_free): simplify conditions (rb_fiber_mark_self): new function (fiber_mark): use rb_fiber_mark_self (cont_save_thread, cont_restore_thread): inline (cont_restore_thread): simplify (fiber_setcontext): simplify conditions (rb_cont_call): remove dereference (fiber_t_alloc): update for rb_fiber_t->prev type change (rb_fiber_start): ditto (fiber_current): extract from rb_fiber_current (return_fiber): move, simplify type checks (rb_fiber_current): use fiber_current (fiber_store): simplify type checks (fiber_switch): ditto, simplify call to fiber_setcontext, use fiber_current (rb_fiber_transfer): update for type changes (rb_fiber_terminate): move, use fiber_switch (rb_fiber_resume): update for type changes (rb_fiber_reset_root_local_storage): ditto (rb_fiber_yield): use rb_fiber_switch instead of rb_fiber_transfer (rb_fiber_m_transfer): ditto [ruby-core:65518] [Feature #10341]

Revision 47964 - 10/15/2014 10:35 PM - normalperson (Eric Wong)

cont.c: Optimize fiber_switch callees

Remove some unnecessary VALUE/struct conversions and aggressively inline
functions used during fiber_switch. Either of these changes alone does
not yield significant performance increase, but in combination they
improve performance by ~6%.

Arguably, removal of separate VALUE/rb_fiber_t* variables also makes the
code more readable in a few places.

- vm_core.h: declare rb_fiber_t typedef (rb_thread_t): fiber and root_fiber become rb_fiber_t * (from VALUE)
- vm.c (rb_thread_mark): use rb_fiber_mark_self
- cont.c (rb_fiber_t): prev becomes rb_fiber_t * (from VALUE) (cont_mark, cont_free): simplify conditions (rb_fiber_mark_self): new function (fiber_mark): use rb_fiber_mark_self (cont_save_thread, cont_restore_thread): inline (cont_restore_thread): simplify (fiber_setcontext): simplify conditions (rb_cont_call): remove dereference (fiber_t_alloc): update for rb_fiber_t->prev type change (rb_fiber_start): ditto (fiber_current): extract from rb_fiber_current (return_fiber): move, simplify type checks (rb_fiber_current): use fiber_current (fiber_store): simplify type checks (fiber_switch): ditto, simplify call to fiber_setcontext, use fiber_current (rb_fiber_transfer): update for type changes (rb_fiber_terminate): move, use fiber_switch (rb_fiber_resume): update for type changes (rb_fiber_reset_root_local_storage): ditto (rb_fiber_yield): use rb_fiber_switch instead of rb_fiber_transfer (rb_fiber_m_transfer): ditto [ruby-core:65518] [Feature #10341]

Revision 47964 - 10/15/2014 10:35 PM - normal

cont.c: Optimize fiber_switch callees

Remove some unnecessary VALUE/struct conversions and aggressively inline
functions used during fiber_switch. Either of these changes alone does
not yield significant performance increase, but in combination they
improve performance by ~6%.

Arguably, removal of separate VALUE/rb_fiber_t* variables also makes the
code more readable in a few places.

- vm_core.h: declare rb_fiber_t typedef (rb_thread_t): fiber and root_fiber become rb_fiber_t * (from VALUE)
- vm.c (rb_thread_mark): use rb_fiber_mark_self
- cont.c (rb_fiber_t): prev becomes rb_fiber_t * (from VALUE) (cont_mark, cont_free): simplify conditions (rb_fiber_mark_self): new function (fiber_mark): use rb_fiber_mark_self (cont_save_thread, cont_restore_thread): inline (cont_restore_thread): simplify (fiber_setcontext): simplify conditions (rb_cont_call): remove dereference (fiber_t_alloc): update for rb_fiber_t->prev type change (rb_fiber_start): ditto (fiber_current): extract from rb_fiber_current (return_fiber): move, simplify type checks (rb_fiber_current): use fiber_current (fiber_store): simplify type checks (fiber_switch): ditto, simplify call to fiber_setcontext, use fiber_current (rb_fiber_transfer): update for type changes (rb_fiber_terminate): move, use fiber_switch (rb_fiber_resume): update for type changes (rb_fiber_reset_root_local_storage): ditto (rb_fiber_yield): use rb_fiber_switch instead of rb_fiber_transfer (rb_fiber_m_transfer): ditto [ruby-core:65518] [Feature #10341]

Revision 47964 - 10/15/2014 10:35 PM - normal

cont.c: Optimize fiber_switch callees

Remove some unnecessary VALUE/struct conversions and aggressively inline
functions used during fiber_switch. Either of these changes alone does
not yield significant performance increase, but in combination they
improve performance by ~6%.

Arguably, removal of separate VALUE/rb_fiber_t* variables also makes the
code more readable in a few places.

- vm_core.h: declare rb_fiber_t typedef (rb_thread_t): fiber and root_fiber become rb_fiber_t * (from VALUE)
- vm.c (rb_thread_mark): use rb_fiber_mark_self
- cont.c (rb_fiber_t): prev becomes rb_fiber_t * (from VALUE) (cont_mark, cont_free): simplify conditions (rb_fiber_mark_self): new function (fiber_mark): use rb_fiber_mark_self (cont_save_thread, cont_restore_thread): inline (cont_restore_thread): simplify (fiber_setcontext): simplify conditions (rb_cont_call): remove dereference (fiber_t_alloc): update for rb_fiber_t->prev type change (rb_fiber_start): ditto (fiber_current): extract from rb_fiber_current (return_fiber): move, simplify type checks (rb_fiber_current): use fiber_current (fiber_store): simplify type checks (fiber_switch): ditto, simplify call to fiber_setcontext, use fiber_current (rb_fiber_transfer): update for type changes (rb_fiber_terminate): move, use fiber_switch (rb_fiber_resume): update for type changes (rb_fiber_reset_root_local_storage): ditto (rb_fiber_yield): use rb_fiber_switch instead of rb_fiber_transfer (rb_fiber_m_transfer): ditto [ruby-core:65518] [Feature #10341]

Revision 47964 - 10/15/2014 10:35 PM - normal

cont.c: Optimize fiber_switch callees

Remove some unnecessary VALUE/struct conversions and aggressively inline
functions used during fiber_switch. Either of these changes alone does
not yield significant performance increase, but in combination they
improve performance by ~6%.

Arguably, removal of separate VALUE/rb_fiber_t* variables also makes the
code more readable in a few places.

- vm_core.h: declare rb_fiber_t typedef (rb_thread_t): fiber and root_fiber become rb_fiber_t * (from VALUE)
- vm.c (rb_thread_mark): use rb_fiber_mark_self
- cont.c (rb_fiber_t): prev becomes rb_fiber_t * (from VALUE) (cont_mark, cont_free): simplify conditions (rb_fiber_mark_self): new function (fiber_mark): use rb_fiber_mark_self (cont_save_thread, cont_restore_thread): inline (cont_restore_thread): simplify (fiber_setcontext): simplify conditions (rb_cont_call): remove dereference (fiber_t_alloc): update for rb_fiber_t->prev type change (rb_fiber_start): ditto (fiber_current): extract from rb_fiber_current (return_fiber): move, simplify type checks (rb_fiber_current): use fiber_current (fiber_store): simplify type checks (fiber_switch): ditto, simplify call to fiber_setcontext, use fiber_current (rb_fiber_transfer): update for type changes (rb_fiber_terminate): move, use fiber_switch (rb_fiber_resume): update for type changes (rb_fiber_reset_root_local_storage): ditto (rb_fiber_yield): use rb_fiber_switch instead of rb_fiber_transfer (rb_fiber_m_transfer): ditto [ruby-core:65518] [Feature #10341]

Revision 47964 - 10/15/2014 10:35 PM - normal

cont.c: Optimize fiber_switch callees

Remove some unnecessary VALUE/struct conversions and aggressively inline
functions used during fiber_switch. Either of these changes alone does
not yield significant performance increase, but in combination they
improve performance by ~6%.

Arguably, removal of separate VALUE/rb_fiber_t* variables also makes the
code more readable in a few places.

- vm_core.h: declare rb_fiber_t typedef (rb_thread_t): fiber and root_fiber become rb_fiber_t * (from VALUE)
- vm.c (rb_thread_mark): use rb_fiber_mark_self
- cont.c (rb_fiber_t): prev becomes rb_fiber_t * (from VALUE) (cont_mark, cont_free): simplify conditions (rb_fiber_mark_self): new function (fiber_mark): use rb_fiber_mark_self (cont_save_thread, cont_restore_thread): inline (cont_restore_thread): simplify (fiber_setcontext): simplify conditions (rb_cont_call): remove dereference (fiber_t_alloc): update for rb_fiber_t->prev type change (rb_fiber_start): ditto (fiber_current): extract from rb_fiber_current (return_fiber): move, simplify type checks (rb_fiber_current): use fiber_current (fiber_store): simplify type checks (fiber_switch): ditto, simplify call to fiber_setcontext, use fiber_current (rb_fiber_transfer): update for type changes (rb_fiber_terminate): move, use fiber_switch (rb_fiber_resume): update for type changes (rb_fiber_reset_root_local_storage): ditto (rb_fiber_yield): use rb_fiber_switch instead of rb_fiber_transfer (rb_fiber_m_transfer): ditto [ruby-core:65518] [Feature #10341]

Revision c36e3466 - 10/16/2014 12:17 AM - nobu (Nobuyoshi Nakada)

cont.c: fix compile error

- cont.c (rb_fiber_t): fix compile error caused by move to vm_core.h at r47964. [Feature #10341]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47969 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 47969 - 10/16/2014 12:17 AM - nobu (Nobuyoshi Nakada)

cont.c: fix compile error

- cont.c (rb_fiber_t): fix compile error caused by move to vm_core.h at r47964. [Feature #10341]

Revision 47969 - 10/16/2014 12:17 AM - nobu (Nobuyoshi Nakada)

cont.c: fix compile error

- cont.c (rb_fiber_t): fix compile error caused by move to vm_core.h at r47964. [Feature #10341]

Revision 47969 - 10/16/2014 12:17 AM - nobu (Nobuyoshi Nakada)

cont.c: fix compile error

- cont.c (rb_fiber_t): fix compile error caused by move to vm_core.h at r47964. [Feature #10341]

Revision 47969 - 10/16/2014 12:17 AM - nobu (Nobuyoshi Nakada)

cont.c: fix compile error

- cont.c (rb_fiber_t): fix compile error caused by move to vm_core.h at r47964. [Feature #10341]

Revision 47969 - 10/16/2014 12:17 AM - nobu (Nobuyoshi Nakada)

cont.c: fix compile error

- cont.c (rb_fiber_t): fix compile error caused by move to vm_core.h at r47964. [Feature #10341]

Revision 47969 - 10/16/2014 12:17 AM - nobu (Nobuyoshi Nakada)

cont.c: fix compile error

- cont.c (rb_fiber_t): fix compile error caused by move to vm_core.h at r47964. [Feature #10341]

Revision 8b1955d0 - 10/16/2014 12:19 AM - nobu (Nobuyoshi Nakada)

cont.c: fix typedef

- cont.c (rb_fiber_struct): remove useless typedef. [Feature #10341]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@47970 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 47970 - 10/16/2014 12:19 AM - nobu (Nobuyoshi Nakada)

cont.c: fix typedef

- cont.c (rb_fiber_struct): remove useless typedef. [Feature #10341]

Revision 47970 - 10/16/2014 12:19 AM - nobu (Nobuyoshi Nakada)

cont.c: fix typedef

- cont.c (rb_fiber_struct): remove useless typedef. [Feature #10341]

Revision 47970 - 10/16/2014 12:19 AM - nobu (Nobuyoshi Nakada)

cont.c: fix typedef

- cont.c (rb_fiber_struct): remove useless typedef. [Feature #10341]

Revision 47970 - 10/16/2014 12:19 AM - nobu (Nobuyoshi Nakada)

cont.c: fix typedef

- cont.c (rb_fiber_struct): remove useless typedef. [Feature #10341]

Revision 47970 - 10/16/2014 12:19 AM - nobu (Nobuyoshi Nakada)

cont.c: fix typedef

- cont.c (rb_fiber_struct): remove useless typedef. [Feature #10341]

Revision 47970 - 10/16/2014 12:19 AM - nobu (Nobuyoshi Nakada)

cont.c: fix typedef

- cont.c (rb_fiber_struct): remove useless typedef. [Feature #10341]

Revision 60473fe0 - 05/08/2017 01:59 AM - normal

benchmark/bm_vm2_fiber_switch.rb: check for fiber performance

There are currently no benchmarks for Fiber performance, I
should've committed this years ago when [Feature #10341] was
implemented.

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@58606 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 58606 - 05/08/2017 01:59 AM - normalperson (Eric Wong)

benchmark/bm_vm2_fiber_switch.rb: check for fiber performance

There are currently no benchmarks for Fiber performance, I
should've committed this years ago when [Feature #10341] was
implemented.

Revision 58606 - 05/08/2017 01:59 AM - normal

benchmark/bm_vm2_fiber_switch.rb: check for fiber performance

There are currently no benchmarks for Fiber performance, I
should've committed this years ago when [Feature #10341] was
implemented.

Revision 58606 - 05/08/2017 01:59 AM - normal

benchmark/bm_vm2_fiber_switch.rb: check for fiber performance

There are currently no benchmarks for Fiber performance, I
should've committed this years ago when [Feature #10341] was
implemented.

History

#1 - 10/08/2014 11:41 PM - normalperson (Eric Wong)

Cool, I can confirm the performance results on one of my systems.
I do not know the fiber code well, but it seems correct.

I think fiber_mark_self (and any non-static functions) needs to be
prefixed with "rb_" even if it is an internal API.

#2 - 10/11/2014 08:58 PM - nome (Knut Franke)

- File 0005-Optimize-fiber_switch-callees.patch added

Attached a revised version of patch 5 with fiber_mark_self replaced by rb_fiber_mark_self. Thanks for the hint.

#3 - 10/12/2014 12:29 AM - normalperson (Eric Wong)

Thanks. I'll wait a few days for others to look and try it out
before committing.

#4 - 10/15/2014 10:34 PM - Anonymous

- Status changed from Open to Closed

- % Done changed from 0 to 100

Applied in changeset r47959.

---

cont.c: Remove unused prev_fiber/next_fiber fields

They were added in r19890 (8a0d53a), with the explanation that it's a
double linked list of fibers in the same thread, but without any code
using them.

- cont.c (rb_fiber_t): remove prev_fiber/next_fiber (fiber_link_join, fiber_link_remove): remove functions (fiber_free, fiber_init, root_fiber_alloc): remove references to removed fields and functions [ruby-core:65518] [Feature #10341]

Author: Knut Franke Knut.Franke@gmx.de

#5 - 10/16/2014 12:15 AM - nobu (Nobuyoshi Nakada)

- Status changed from Closed to Open

I've missed this ticket, and r47964 caused compile error.

#6 - 10/16/2014 12:17 AM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Closed

Applied in changeset r47969.

---

cont.c: fix compile error

- cont.c (rb_fiber_t): fix compile error caused by move to vm_core.h at r47964. [Feature #10341]

#7 - 10/16/2014 04:29 AM - ko1 (Koichi Sasada)

- Status changed from Closed to Open

on mswin32, the following simple script doesn't work.

```
p Fiber.new{
  100
}.resume

#=>
C:/ko1/src/ruby/trunk/test.rb:1:in `p': method `inspect' called on
hidden T_OBJECT object (0x1ea6730 flags=0x1) (NotImplementedError)
        from C:/ko1/src/ruby/trunk/test.rb:1:in `<main>'
NMAKE : fatal error U1077: '.\miniruby.exe' : ▯▯▯▯ ▯▯▯ '0x1'
Stop.
```

#8 - 10/16/2014 04:30 AM - ko1 (Koichi Sasada)

disable FIBER_USE_NATIVE works fine on mswin32.

#9 - 10/16/2014 05:10 AM - normalperson (Eric Wong)

Maybe this is a simple fix for win32, but I cannot test:

diff --git a/cont.c b/cont.c
index 739ec80..08acf40 100644
--- a/cont.c
+++ b/cont.c
@@ -1366,10 +1366,10 @@ fiber_store(rb_fiber_t *next_fib, rb_thread_t *th)*
terminated_machine_stack.ptr = NULL;
terminated_machine_stack.size = 0;
}
+#endif / not _WIN32 */*
fib = th->fiber;
if (fib->cont.argc == -1) rb_exc_raise(fib->cont.value);
return fib->cont.value;
-#endif / not _WIN32 */

#else /* FIBER_USE_NATIVE */
cont_save_machine_stack(th, &fib->cont);

#10 - 10/16/2014 05:58 AM - ko1 (Koichi Sasada)

Eric Wong wrote:

> Maybe this is a simple fix for win32, but I cannot test:

Thank you. Now, it is working. I'm running test-all now.

#11 - 10/16/2014 07:20 AM - ko1 (Koichi Sasada)

It works fine!

Thank you.

#12 - 10/18/2014 11:20 AM - nome (Knut Franke)

D'oh. Thanks for fixing the win32/non-native issues.

And great to see this included. :-)

#13 - 05/08/2017 01:59 AM - Anonymous

- Status changed from Open to Closed

Applied in changeset trunk|r58606.

---

benchmark/bm_vm2_fiber_switch.rb: check for fiber performance

There are currently no benchmarks for Fiber performance, I should've committed this years ago when [Feature #10341] was implemented.

Files

0001-Remove-unused-prev_fiber-next_fiber-fields.patch	2.02 KB	10/08/2014	nome (Knut Franke)
0002-Code-cleanup-in-fiber_switch-fiber_store.patch	4.79 KB	10/08/2014	nome (Knut Franke)
0003-Small-code-cleanup.patch	1.16 KB	10/08/2014	nome (Knut Franke)
0004-cont_save_thread-Sparse-copying-of-thread-data.patch	3.39 KB	10/08/2014	nome (Knut Franke)
0005-Optimize-fiber_switch-callees.patch	13.9 KB	10/08/2014	nome (Knut Franke)
0005-Optimize-fiber_switch-callees.patch	13.9 KB	10/11/2014	nome (Knut Franke)