Ruby master - Feature #13378

Eliminate 4 of 8 syscalls when requiring file by absolute path

03/28/2017 07:46 PM - burke (Burke Libbey)

Status:	Open
Priority:	Normal
Assignee:	nobu (Nobuyoshi Nakada)
Target version:	

Description

Don't open file twice when specified by absolute path.

When invoking require '/a.rb' (i.e. via an absolute path), ruby generates this sequence of syscalls:

```
open    /a.rb
fstat64 /a.rb
close   /a.rb
open    /a.rb
fstat64 /a.rb
fstat64 /a.rb
read    /a.rb
close   /a.rb
```

It is apparent that the only inherently necessary members of this sequence are:

```
open    /a.rb
fstat64 /a.rb
read    /a.rb
close   /a.rb
```

(the fstat64 isn't *obviously* necessary, but it does serve a purpose and probably shouldn't be removed).

The first open/fstat64/close is used to check whether the file is loadable. This is important when scanning the $LOAD_PATH, since it is used to determine when a file has been found. However, when we've already unambiguously identified a file before invoking require, this serves no inherent purpose, since we can move whatever work is happening as a result of that fstat64 into the second open/close sequence.

This change bypasses the first open/fstat64/close in the case of an absolute path to require. It also removes one of the doubled-up fstat64 calls later in the sequence. As a result, the number of syscalls to require a file changes:

- From 8 to 4 when specified by absolute path;
- From 5+3n to 4+3n otherwise *(where n is the number of $LOAD_PATH items scanned)*.

In future work, it would be possible to re-use the file descriptor opened while searching the $LOAD_PATH without the close/open sequence, but this would cause some ugly layering issues.

We intend to use this in conjunction with something like https://github.com/shopify/bootscale*, which pre-resolves required features to absolute paths before calling require. This change reduces our total number of filesystem accesses by 13% during application boot.*

Various notes and rationale at http://notes.burke.libbey.me/ruby-require-optimization

History

#1 - 03/28/2017 08:31 PM - burke (Burke Libbey)

- File 0001-reduce-syscalls-on-require-fixed.patch added

Ah, I was too hasty with the rebase from my 2.3.3 branch. I've attached a fixed patch. Note also that trunk has already eliminated the double-fstat, so this only reduces the number of syscalls when a feature is specified by absolute path (from 7 to 4).

#2 - 05/26/2017 06:32 PM - naruse (Yui NARUSE)

```
-VALUE rb_find_file_safe(VALUE, int);
+VALUE rb_find_file_safe(VALUE, int, int);
```

When you are CRuby developer, all functions declared under include/*/.h are considered public C API.
(Note that if you are C extension developer, some of such APIs are experimental or private...)

Therefore rb_find_file_safe() should be kept as is.
You should declare rb_find_file_safe_with_defer_load_check or something in internal.h (not include/ruby/intern.h).

Or no one seems to use the function by GitHub search
https://github.com/search?utf8=%E2%9C%93&q=rb_find_file_safe++extension%3Ac+path%3Aext&type=Code
we can simply change the prototype and move to internal.h.

```
            if ((fd = rb_cloexec_open(fname, mode, 0)) < 0) {
-               rb_load_fail(fname_v, strerror(errno));
+               goto fail;
             }
```

This seems to need e = errno; before goto fail;

```
    int fd = 0;
```

It should be -1 because 0 is still valid fd even though it is STDIN.

```
  if (fd > 0) (void)close(fd);
```

ditto.

Additionally, if you have tests to confirm the behavior, could you add them to test/ruby/test_require.rb?

#3 - 05/29/2017 06:56 PM - burke (Burke Libbey)

- File 0001-reduce-syscalls-on-require-v2.patch added

Thank you for the feedback! I've attached an updated patch to address the issues.

As for testing it, I haven't been able to think of a reasonable method to verify the behaviour without using dtrace/strace, since the only observable effect without system-level instrumentation should be a slight reduction in runtime. I *could* add a case to DTrace::TestRequire, but it feels wrong, since this file is concerned with testing the dtrace probes implemented by ruby, not testing ruby using built-in dtrace probes.

Suggestions?

#4 - 06/16/2017 07:57 AM - ko1 (Koichi Sasada)

- Assignee set to nobu (Nobuyoshi Nakada)

Files

0001-reduce-syscalls-on-require.patch	7.56 KB	03/28/2017	burke (Burke Libbey)
0001-reduce-syscalls-on-require-fixed.patch	6.94 KB	03/28/2017	burke (Burke Libbey)
0001-reduce-syscalls-on-require-v2.patch	6.44 KB	05/29/2017	burke (Burke Libbey)