从 Lua 5.2 迁移到 5.3

在 2015 年的新年里，Lua 5.3 发布了 rc3 版。

如果回顾 Lua 5.2 的发布历史，Lua 5.2 的 final 版是在 rc8 之后的 2011 年 12 月 17 日发布的，距离 rc1 的发布日 2011 年 11 月 24 日过去不到 1 个月。我们有理由相信正式版不远了。(5.3 的 rc1 是 2014 年 12 月 17 日发布的）

这次升级对 Lua 语言层面的影响非常的小，但新增加的 int64 支持，以及 string pack 、utf8 库对开发帮助很大。所以我强烈建议正在使用 Lua 5.2 的项目尽快升级到 5.3 。相对而言，当初 5.1 向 5.2 升级的时候就痛苦的多（去掉了 setfenv ，增加了 _ENV）。

我计划在 Lua 5.3 正式发布后，将 skynet 内置的 Lua 版本升级到 5.3 ，然后着手进行 skynet 1.0 的发布工作。

在 skynet 的应用环境下，我还是需要对 lua vm 的实现打一个 patch 让 不同的 lua vm 间可以共享 Proto 。但这个工作可以先不忙做，等正式发布后再来也可以。

目前可以先逐步升级 skynet 下的 lua 库。

我已经在 github 项目下创建了一个叫 lua53 的分支，做了一些工作。希望有同学可以帮忙一起 review 这部分代码。有兴趣的同学可以对照 最新的 commits 来检查这些升级做的变更。

必须做的修改是去掉 unsigned 有关的 api 调用。

lua 5.3 去掉了 lua_pushunsigned lua_tounsigned 等 api ，现在一律使用 lua_pushinteger 等。这些 api 默认操作 lua_Integer 这个数据类型。按文档的说法，在你的代码中，应该尽可能的使用 lua_Integer 。它默认等价于 long long ，至少保证 64 位字长（lua 5.3 可以配置成使用 32bit 整数，但在 skynet 的应用环境不会这么做）。如果需要无符号整数，可以再在 C 代码中做强制类型转换。

这部分工作做完后，整个代码就可以正确编译了。

但是，和序列化有关的库还需要为 lua 5.3 优化。因为 lua 5.3 原生支持了整型，不需要全部转换成 double 类型储存数字。

之前在做数据序列化工作时（seri 库 和 bson 库等），为了区分一个 number 类型到底是浮点数还是整数，我采取的方法是用 lua_tonumber 和 lua_tointeger 分别取一次，然后比较两个数值是否相等。在 lua 5.3 中，直接提供了更高效的 lua_isinteger 来做判断。

由于现在直接支持 64bit 整数，就不再需要使用 lightuserdata 来保存长整数了。所以我去掉了 int64 库 。

相应的，相关的库应该做一些调整。pbc 库目前没有打包在 skynet 项目中，但我已经修改完毕，晚一点再放出来。skynet 内自带的序列化库，以及 bson ， redis 都需要做一些调整。

btw, 再修改序列化库时发现一个 bug ，再不支持非对齐地址访问的架构下会有点问题，这次一并修改了。

这次，lua 5.3 中把 __ipairs 去掉了，并且重写了 table 库。为了 Conceptual Integrity ，而敢于删改过去的东西，一直是我很欣赏 lua 的地方。

lua 5.3 同样去掉了 bit32 库（打开 5.2 兼容模式时，这个库还是存在的），而且这个库只对 32bit 整数有效，位操作现在提供了原生的操作符支持。（注：xor 是用 ~ 而不是 ^ ，因为 ^ 已经被用于 pow 操作了）我检索了整个代码，发现用到 bit32 最多的是那个从 openresty 移植来的 mysql driver 。

但实际上，在 lua 5.3 中不必再使用位操作去解析数据流了。因为有新的 string.pack 这个强大的 api 。比如:

```
local function _get_byte8(data, i)
    local a, b, c, d, e, f, g, h = strbyte(data, i, i + 7)

    -- XXX workaround for the lack of 64-bit support in bitop:
    local lo = bor(a, lshift(b, 8), lshift(c, 16), lshift(d, 24))
    local hi = bor(e, lshift(f, 8), lshift(g, 16), lshift(h, 24))
```

```
    return lo + hi * 4294967296, i + 8
end
```

这个函数可以被简化成：

```
local function _get_byte8(data, i)
    return strunpack("<I8",data,i)
end
```

在修改过程中，我发现 openresty 里这块代码写的很不 lua ，比如这个 dump 函数，

```
local function _dump(data)
    local len = #data
    local bytes = new_tab(len, 0)
    for i = 1, len do
        bytes[i] = format("%x", strbyte(data, i))
    end
    return concat(bytes, " ")
end
```

按 lua 的惯用法应该写成：

```
local function _dump(data)
    return string.gsub(data, ".",
        function(x) return format("%02x ", strbyte(x)) end)
end
```

这样既简洁，性能也好很多。

其实这是个普遍的问题。由于 Lua 天生是门嵌入语言，几乎所有的 Lua 程序员都用过别的语言。所以许多 Lua 程序员带着其他语言的经验来写。前段时间我就发现过另一个例子。

由于 mysql 这块改动最多，所以特别需要有人来一起 review 和测试。当然这块代码还有很多可以改进的地方，暂时就没有精力做了。如果有同学有兴趣，还可以把那块尚未完成的编码设置加进去。

云风 提交于 January 6, 2015 11:36 AM | 固定链接

COMMENTS

@agentzh

我随机生成了不同长度的 40K 个字符串, 在 luajit 2.0 下做了对比测试.

初步结论是: 在关闭 gc 时, 两个版本的运行时间几乎是相同的. (3% 的浮动范围内)

但是如果生成临时 table 的话, 会多消耗 8 倍的临时内存. (倍数取决于字符串长度)

如果考虑 gc 的影响, 就不太好做简单的比较了.

Posted by: Cloud | (8) January 7, 2015 03:18 PM

@agentzh

那个 closure 在 lua 5.2 中其实不需要动态创建的, 因为它不依赖任何上下文相关的 upvalue . 这个 closure 是被 cache 的. 如果罗嗦一点的话, 可以事先定义好, 再传进去.

我觉得单就这个函数而言, gsub 函数的性能不可能低于 jit 生成的代码. 除非 gsub 本身的实现有问题.

当然, 我主要说的是 gsub 是 lua 语言的惯用法. 遵循惯用法在我看来比性能更重要.

其实我谈的性能问题不光是时间性能, 最主要的差别的空间性能. 我不太清楚 jit 对临时生成的 table 有没有特别的优化.

Posted by: Cloud | (7) January 7, 2015 03:06 PM

作为 openresty 中那个 _dump 函数的作者，我需要指出的是那样写纯粹是为了那段代码能被 LuaJIT 2.1 完全地 JIT 编译。你的版本无法被 JIT 编译，因为用了 string.gsub() 和动态 closure 创建（即 function ()...end）。

当然了，这个 _dump 函数纯粹是为了内部调试目的，所以如何写倒也无所谓了，呵呵，毕竟生产上不会走这个函数 :)

根据我自己的 benchmark，是否被 JIT 编译对于 lua-resty-mysql 这个库来说，性能差别还是挺大的。

Posted by: agentzh | (6) January 7, 2015 02:52 AM

openresty那么写是因为gsub不能被jit，人家主要追求的是速度

Posted by: heeroz | (5) January 7, 2015 12:20 AM

最近比较了lua 5.1, 5.2, 5.3的运行性能, 结果是越来越差, 很令人失望, 尤其是5.3比5.2差得非常明显, 可能是64位整数和浮点数之间的判断、转换和混合运算的开销.

Posted by: dwing | (4) January 6, 2015 04:35 PM

早就分裂了， 我已经无视 luajit 了。

Posted by: Cloud | (3) January 6, 2015 03:51 PM

但是luajit還停留在不完全兼容5.2的狀態，感覺lua已經分裂成兩個語言了。

Posted by: larme | (2) January 6, 2015 02:58 PM

冲着string.pack我也会升级，谢谢云风

Posted by: 吴友仁 | (1) January 6, 2015 02:39 PM

POST A COMMENT

非这个主题相关的留言请到：留言本

名字：

Email 地址：

为了验证您是人类，请将六加一的结果（阿拉伯数字七）填写在下面：

URL:

记住我的信息？

留言：
（不欢迎在留言中粘贴程序代码）

提交