Vol.2012-IOT-18 No.6
2012/6/28

組み込みスクリプト言語 mruby を利用した Web サーバの機能拡張支援機構

松本亮介[†]　　岡部寿男[††]

Web サーバを利用したサービスの増加に伴って，Web サーバ上で生じるインシデントも増加している．これらの問題を解決するために，Web サーバソフトウェアの機能拡張が必要である．Apache を利用していた場合，機能拡張のためには C 言語や Apache 内部の仕様を深く理解している必要があり，開発の敷居が高く，コンパイルが必要となって保守性が低い．そこで，現在注目されてきている組み込みスクリプト言語 mruby を利用して，Apache に機能拡張のための mruby 用インターフェイスを実装し，mruby スクリプトによって簡単に Apache 内部の機能拡張を行える機構 mod_mruby を提案する．mod_mruby は，Apache 内部に存在する各種リクエスト処理フェーズにおいて，任意の mruby スクリプトを任意のフェーズでフックして実行できるインターフェイスを提供している．フックする際には，Apache 内部でのみ保持しているリクエスト処理情報を，組み込みスクリプトの特性を生かして，mruby スクリプト上で操作可能にした．また，mruby スクリプト実行時に生成される状態遷移を保存する領域や拡張ライブラリのロードを複数の mruby スクリプトで共有し，コンパイルされたバイトコードのみを使い分ける事によって，高速に動作するように設計した．その結果，mruby は広く利用されている Ruby と同様のオブジェクト指向による実装が可能である事と相まって，多くの開発者が Apache の機能拡張に取り組み易くなると考えている．

A Web Server Extension Mechanism Using Embeddable Scripting Language mruby

Ryosuke Matsumoto[†]　and Yasuo Okabe[††]

As the increase of services using Web servers, the number of incidents also is increasing rapidly. In order to solve those problems, it is necessary to extend a functionality of a Web server software. In case of using Apache, developers are required high coding skill of C language and internal specifications of Apache in order to extend the functionality of it. The development of a web server extension requires some high skills, and the maintainability is low since that extension need to compile a code. Therefore, we propose mod_mruby that is a web server extension mechanism using embeddable scripting language mruby which has been attracting attention now. mod_mruby allows to extend the functionality of Apache easily by implementing a mruby script. mod_mruby provides an interface to hook and execute any mruby scripts in the various phases of processing requests inside Apache. When hooking mruby scripts, mruby scripts can process the data of processing requests inside Apache, taking advantage of the characteristics of a embeddable scripting language for C language. We have designed that mod_mruby run at high speed by sharing the data of state transition and the extension library of mruby by multiple mruby scripts and using only different byte code each mruby script. Many developers can implement a web server extension easily by mod_mruby in cooperation with coding style of mruby which is the same as object oriented programming ruby which is widely used by web developers.

1. はじめに

クラウドコンピューティングを個人だけでなく企業でも利用する機会が増大している．これまで社内にシステムを保持していた企業は，自社のシステムを，Web サービスを介して利用するようになってきている．それに伴い，Web サービスは日々高度化してきており，システム連携に Web API[2]を活用するケースが増加してきている．その結果，Web サーバの利用頻度は非常に高くなり，より重要なシステム基盤になってきている．その一方で，Web サーバ上で生じるサーバ高負荷問題や，セキュリティインシデントも増加している．そこで，我々は大規模ホスティング環境で生じる高負荷問題や運用技術を改善するために，汎用性の高い大規模共有型 Web バーチャルホスティング基盤のセキュリティと運用技術の改善[3]を，Web サーバソフトウェアの機能拡張を行う事で実現した．このように，Web サービスの高度化に追随するためには，Web サービスのコンテンツのチューニングだけでなく Web サーバソフトウェア自体の内部機能拡張を行う必要が生じてきている．

現状，最もシェアが高い Web サーバソフトウェアは，Apache HTTP Server[4]（以降 Apache とする）である．Apache は最低限の機能をコアとして持ち，その他の機能は Apache モジュールを追加する事で機能拡張する方式をとっている．しかし，Apache モジュール開発の情報は少なく，簡易機能を追加するだけでも，C 言語や Apache 内部の仕様を深く理解している必要があり，開発の敷居が高い．また，モジュール組み込み時にも，コンパイルや Apache の再起動が必要となり，保守性が低い．そこで，これまで，スクリプト言語の Perl や Ruby で，Apache モジュール相当の実装を可能とする mod_perl や mod_ruby が開発されてきた．しかし，ライブラリが数多く組み込まれた言語自体のリソース使用

† 京都大学 情報学研究科
Graduate School of Informatics, Kyoto University
†† 京都大学 学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University

量が非常に大きく，静的ファイルのリクエストを含め，Apache の内部処理として常に呼び出される処理としては，あまりにも無駄が多いと考えられる．最近では，組み込みスクリプト言語 Lua[5]を Apache に組み込む方式 mod_lua も開発されている．しかし，軽量組み込むスクリプトであるため，リソース使用量は少なくなったものの，内部処理を扱うための機能が十分でないために柔軟性が低く，処理速度も言語の高速性を生かす実装になっていない．

そこで，サーバソフトウェアの内部処理として実行されるスクリプトにおいて，処理の高速性，少ないリソース消費量，内部機能拡張の柔軟性，及び，スクリプト言語による開発や保守効率の向上に着目し，それらを同時に実現するための Apache の機能拡張支援機構を提案する．実装には，現在注目されてきている組み込みスクリプト言語 mruby[6]を採用した. 開発した Apache の機能拡張支援機構は mod_mruby と呼ぶ．mod_mruby は Apache に機能拡張専用の mruby 用インターフェイスを提供しており，mruby スクリプトによって容易に Apache 機能拡張を行うことが可能である．mod_mruby によって，Apache 内部に存在する各種リクエスト処理フェーズにおいて，任意の mruby スクリプトを任意のフェーズでフックして実行し，Apache の内部機能として処理する事ができる．

mod_mruby では，処理の高速性を生かすために，mruby スクリプト実行時に生成される状態遷移保存領域の確保や拡張ライブラリのロードを Apache のサーバプロセス上で複数の mruby スクリプトによって共有し，コンパイルされたバイトコードのみを使い分ける事によって，高速に動作するように設計した. さらに, リソース消費量の観点では，mruby が軽量組み込みスクリプトであるため，スクリプト言語によるリソースの使用量を非常に少なくする事が可能である．また，mruby 上で内部機能を柔軟に拡張できるように，C 言語と連携が容易な組み込みスクリプトの特性を生かして，Apache の内部と処理やデータを効率良く連携するための API を設計中である．mruby スクリプトを変更する事で，容易に Apache の機能を変更及び拡張が可能となり, 面倒なコンパイルや Apache の再起動が必要無くなる．さらに，他の Web サーバソフトウェア Nginx[7] に mod_mruby 相当の機能を実装し，Web サーバ用の mruby ライブラリを組み込む事で，Web サーバソフトウェア間の実装の違いを mruby ライブラリで吸収し，Web サーバ拡張機能を mruby での実装に統一できると考えている．

本稿では，Web サービスの高度化を考慮して，開発者が Web サーバソフトウェアの機能拡張を行う際に，軽量スクリプト言語 mruby を用いて，Web サーバソフトウェア機能を容易に追加実装可能な機能拡張支援機構 mod_mruby を提案する．2 章では Apache モジュールとサーバ機能拡張支援について述べる．3 章では mod_mruby のアーキテクチャと、現状の mruby スクリプトによる機能拡張方法について説明する. 4 章で mruby スクリプトを Apache の内部機能として実行した場合のパフォーマンス評価を行い，5 章でむすびとする．

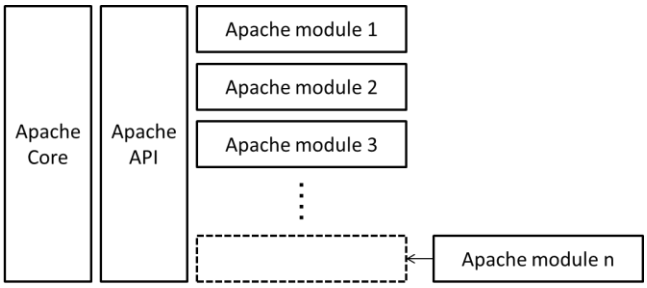

図 1　Apache モジュールの仕組み

2. Apache モジュールとサーバ機能拡張支援

Apache は Web サーバソフトウェアであり，世界で最もシェアの高いソフトウェアである. Apache の特徴としては，最低限の Web サーバ機能をコアとして持ち，その他の機能はモジュールとして追加できるような設計をとっている．図 1 に, Apache のコアとモジュールの概要図を示す. Apache のコアと Apache モジュールの連携は，Apache 独自の API を介して実現されている．モジュールの実装においては，制限はほとんどなく，コアに近い実装から Web アプリケーションに近い実装まで，様々な領域の実装が可能となっている．Apache モジュールは C 言語で実装し，Apache 内部で直接 Apache モジュール内の関数をフックする．Perl や PHP，Python，Ruby 等に代表されるようなスクリプト言語で実装する Web コンテンツにおいて，単純な処理であれば，Apache モジュールで実装できる. 高速に動作する事を優先した場合は，Apache モジュールでアプリケーションを開発する事も可能である。

しかし，一方で，Apache モジュールは C 言語で実装する必要があるためにコンパイルが必要である．また，Apache に組み込む必要があり，Apache デーモンの再起動が必要となるため，開発効率や保守性の面でスクリプト言語より劣る．さらに，Apache モジュール開発のドキュメントは少なく，特に日本語のドキュメントは殆どないため，開発の敷居が高くなっている. そこで, 過去にスクリプト言語の Perl や Ruby で，Apache モジュール相当の実装を可能とする拡張支援機構が開発されてきた．しかし，それらの機能は，スクリプト言語を Web コンテンツとして扱う用途の付加的な機能として実装されているに過ぎなかった. そのため，Apache の内部処理としてスクリプトをフックする機能としては，ライブラリが数多く組み込まれた言語自体のメモリ使用量やリソース使用量が非常に大きく，静的ファイルのリクエストを含め，Apache の内部処理として常に呼び出される処理としては，あまりにも無駄が多いとされ，ほとんど普及しなかった.

そのような背景の中，軽量組み込みスクリプト言語である Lua が人気を高めてきた．Lua はリオデジャネイロ・カトリカ大学の情報工学科コンピュータグラフィックステクノロジーグループ TeCGraf によって設計開発されたスクリプト言語である．Lua という名前はポルトガル言語で月を意味する．Lua は C 言語のホストプログラムに組み込まれる事を目的に設計されており，高速な動作と高い移植性，組み込みの容易さが特徴である．移植性を高めるため，一旦バイトコードにコンパイルされ，Lua VM 上で実行する方式をとっている．変数に型のないスクリプト言語では最速の言語・処理系だとされている．

Lua を Apache でも利用できるような Apache モジュールの開発が行われてきている．2012 年 4 月 20 日に数年ぶりの Apache のメジャーバージョンである 2.4.1 がリリースされた．Apache2.4.1 は，シェアを伸ばしてきている Nginx に対抗するために，各種パフォーマンス改善や新機能の実装が行われた．新機能の中には，Lua を Apache でも利用できるように，Apache に Lua を組み込むためのモジュール mod_lua が実験的に導入されている．mod_lua によって，Lua スクリプトを Apache の Web コンテンツとして扱う事ができ，また，Apache の内部処理として，Lua スクリプトで定義した関数を呼び出す事が可能になっている．Apache の内部処理として Lua スクリプトを呼び出せば，スクリプトとしての保守性や開発効率を生かしたまま，これまでの Perl や Ruby よりも軽量かつ高速に Apache の内部処理を実装する事が可能となる．しかし，mod_lua は Web コンテンツとして扱う用途と Apache の内部処理をスクリプトで実装する用途を目的としているため，幾つかの問題がある．第一に，Lua との仕様と相まって，Lua スクリプト実行時に，状態遷移を保存する領域をスクリプト単位で確保する必要があり，パフォーマンスが低下する．状態遷移保存領域は，スクリプトを実行する上で必要な情報を格納する用途で使われ，確保のコストが非常に高い．Lua の仕様においては，一つの状態遷移保存領域で，複数のスクリプトを共有する事が可能であるが，共有してしまうと，それぞれのスクリプトから共有している他のスクリプトのファンクションを呼び出せてしまう．このような実装をとってしまうと，Web コンテンツで扱った場合や，複数の Apache の内部処理を実装した場合，スクリプト上のファンクションが干渉し合う可能性が高いため，mod_lua では共有していないと考えられる．第二に，Apache の内部処理を扱う機能が十分でない事が挙げられる．Apache の内部処理を実装するために必要な，Apache 内部で扱われるリクエストや設定情報を保存した構造体や，複数存在する各種フックフェーズを Lua 上で扱うためのライブラリが十分でない．例えば，Apache でログを出力するためのフックフェーズである ap_hook_log_transaction()で Lua スクリプトを呼び出す事ができない．また，Apache でリクエストを受けてからレスポンスを返すまでに扱われる情報を格納した request_rec 構造体のデータにおいて，扱えるデータが少なかい．そのため，簡易な内部処理しか実装できないようになっている．第三に，Lua 言語が，Apache や middleware，及び，OS を扱う技術者にとってあまり馴染みのない言語である事が挙げられる．Apache のモジュールの機能拡張を支援する事を目的とした場合，Web に関わる技術者にとって馴染みのある Perl や Ruby，PHP のような言語で実装できた方が普及し易いと考えている．

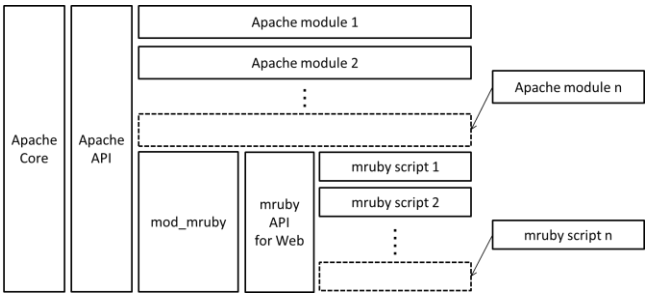

図 2　Apache と mod_mruby の仕組み

3. mod_mruby のアーキテクチャと適応例

組み込みスクリプト言語 mruby を利用して，Apache に機能拡張専用の mruby 用インターフェイスを実装し，mruby スクリプトによって容易に Apache 機能拡張を行える機構 mod_mruby を提案する．mruby は，2012 年 4 月 20 日にソースが公開された組み込みスクリプト言語である．高い生産性を提供するために Ruby と同様のオブジェクト指向による記述はそのままに，様々な組み込み用途でも利用可能なスクリプト言語である．mruby は Lua と同様，軽量かつ高速に動作する事を目的としている．なお，mruby は現在も開発中で，正式リリースはまだ先になると思われるので，言語自体の細かな仕様に関して本稿では言及しない．mod_mruby は，Apache 内部に存在する各種リクエスト処理フェーズにおいて，任意の mruby スクリプトを任意のフェーズでフックして実行できるインターフェイスを提供している．Web コンテンツとして扱う用途はスコープの範囲外とした．Web コンテンツとして扱う場合は，言語の軽量さよりも言語やライブラリの充実さが重要だと考えており，そのような用途においては，現行の Ruby や PHP 等の方が，より Web コンテンツ開発には適していると考えたためである．それによって，なるべく実装をシンプルにし，無駄な機能を省略する事で，軽量なモジュールにできる．

これまでは，Apache のモジュールを C 言語で実装後，コンパイルしてから Apache に組み込む必要があった．その場合は，Apache の再起動が必要となる．また，保守においては再度コンパイル環境で変更してコンパイル後，再度組み込む必要があった．しかし，mod_mruby は，予め Apache と mruby スクリプトの処理を連携するインターフェイスを

Apache モジュールとして実装しておく. 図 2 に, mod_mruby の仕組みを示す．mod_mruby を Apache に組み込む事によって, mruby スクリプトに Apache の内部処理を実装できるようになる．mruby と Apache は専用の mruby 用 API ライブラリを介して連携する．mruby 上で記述できないような複雑な処理の場合は, Apache モジュールとして組み込む事も可能である. また, mruby スクリプト内の実装の変更も，スクリプトを書き換える事で，即時 Apache の内部処理として反映する事ができる．

2 章で，mod_lua の性能に関する実装の問題を指摘した．図 3 に mod_lua のアーキテクチャを示す．mod_lua においては，スクリプト単位で状態遷移を保存する領域を確保するため，ライブラリの読み込みもスクリプト単位で必要となる．しかし，mod_mruby においては，状態遷移保存領域を複数のスクリプトで共有するようにし，ライブラリの読み込みを複数回実行する必要の無いアーキテクチャを設計した．図 4 に mod_mruby のアーキテクチャを示す．まず，Apache の仕様上，起動時に子サーバプロセス（スレッド）を複数プールさせて起動する．プールされた子サーバプロセスの初期化フェーズである，ap_hook_child_init()の際に，mruby の状態遷移の保存領域である mrb_state 構造体をプロセス単位に mrb_open()によって生成しておく．また，同時に，mod_mruby が提供するインターフェイスにおいて，mruby スクリプト上で扱える Apache の内部処理を操作するための関数ライブラリ群を状態遷移の保存領域に読み込んでおく．そして，mruby スクリプトが Apache からフックされると，状態遷移保存領域の確保やライブラリ読み込みは省略し，子サーバプロセス初期化時に生成しておいた状態遷移保存領域を複数の mruby スクリプトで共有する．そして，mruby スクリプト自体の構文木解析を mrb_parse_file()によって行い, mrb_generate_code()によってバイトコードに変換する．そのバイトコードのみを，状態遷移保存領域内に存在するバイトコードを保持しておくための irep テーブルに格納しておく．このようなアーキテクチャを取ることで, Apache のように大量のリクエストが発生して，mruby スクリプトが Apache からフックされ，その回数だけコンパイルされるような状況において，状態遷移保存領域を複数回確保しなくてよい．その結果，スクリプト単位で状態遷移保存領域の確保が必要である mod_lua と比較して, 高速かつ効率良く処理を行う事ができる. また，mruby は Lua と異なり，状態遷移保存領域である mrb_state を共有した場合，irep テーブルにそれぞれのスクリプトのコンパイルされたバイトコードを格納していく実装をとっている．そのため，mruby の組み込みを扱う C 言語上の実装で，mruby スクリプトから irep テーブルへアクセスのためのインターフェイスを用意しない限り，基本的にはスクリプト間で干渉は生じないと考えられる．

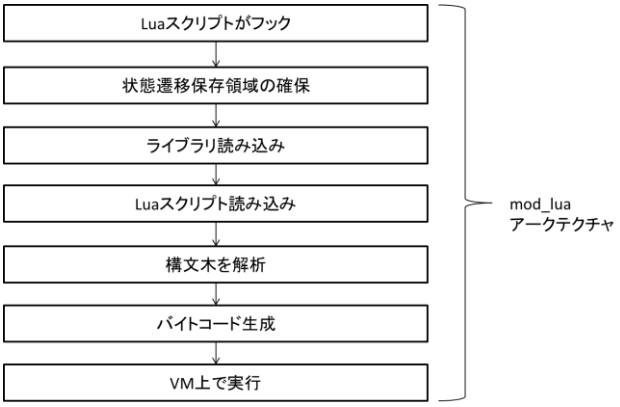

図 3　mod_lua のアーキテクチャ

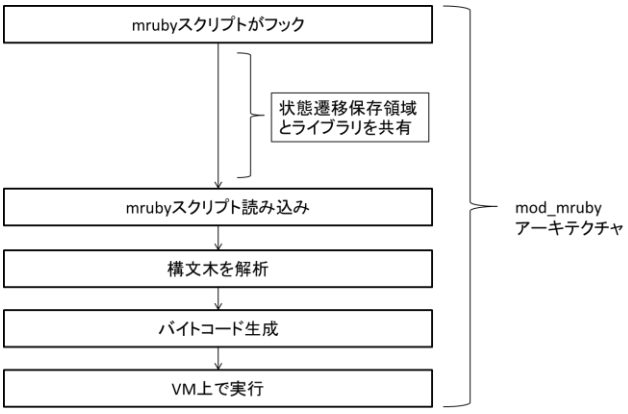

図 4　mod_mruby のアーキテクチャ

```
LoadModule mruby_module    modules/mod_mruby.so
mrubyTranslateNameMiddle   /path/to/mapper.mrb
```

図 5　mod_mruby の Apache 設定例

```
require “Apache”

r = Apache::Request.new()
r.filename = "/var/www/html/redirect.html“

Apache.return(Apache::OK)
```

図 6　mruby スクリプト例

現状，mod_mruby によって Apache 内部の処理を mruby スクリプトで実装するために，幾つかのライブラリを用意している．ライブラリに関しては，まだ開発段階であり，十分には用意されていないが, syslog や Apache の error_log に出力するためのメソッドや, Apache がリクエスト受けてレスポンスを返すまでに，様々な設定情報や処理情報を格納する request_rec 構造体を操作できるインターフェイスを提供している．例えば，Apache がリクエストを受けて，アクセスのあった URI とファイルパスを紐づける

ap_hook_translate_name フェーズにおいて， URI と任意のファイルパスを紐づけて，レスポンスを返す処理を記述した mruby スクリプトを，Apache からフックする場合の例を示す．図 5 は Apache の設定例，図 6 は mruby スクリプトの記述例である．また，図 7 に，request_rec 構造体の各種メンバの値を取得してクライアントに表示する例を示す．

mruby は組み込みスクリプトであるため，スクリプト言語によるリソースの使用量は非常に少なく，高速に動作する．さらに，mruby は Ruby と同様のオブジェクト指向による実装が可能なため，現在普及しているスクリプト言語 Ruby を使って Web アプリケーションを実装したり，ミドルウェアを実装したりしている技術者が，Apache の機能拡張に取り組み易くなると考えている．

4. mod_mruby のパフォーマンス評価

現状の mod_mruby が実用に耐えうるかを評価するために，Apache の内部処理を実装した mruby スクリプトを，mod_mruby を介してフックした場合の性能を評価した．クライアントがどのような URI にアクセスしても，全てのアクセスに対して hello world の文字列を返す処理を実装した．評価においては，同様の処理を C 言語によって実装した Apache モジュール mod_hello と，mod_lua を介して Lua で実装した Lua スクリプトとの比較を行った．

表 1 は，クライアントとサーバマシンの性能を示している．クライアントマシンから，サーバマシンに対して，ab コマンドにより，同時接続数 100，総接続数 3000 でアクセスし，サーバマシンが 1 秒間に処理できたレスポンス数をそれぞれ 10 回計測した．mod_mruby を介して mruby スクリプトを読み込む処理が，C 言語で実装した Apache モジュールと比べてどの程度性能劣化が生じるかを算出した．同時接続数のパラメータは，サーバの Apache や OS のチューニングがボトルネックにならないように，予備実験により適切だと思われるパラメータを設定した．

図 8 は，10 回行った実験結果のグラフを示している．横軸はテスト回数，縦軸は 1 秒間にサーバが処理可能であったレスポンス数を表している．実験の結果，mod_mruby が mod_lua よりも性能が非常に高い事が分かった．これは，状態遷移保存領域を，mod_lua はスクリプト実行毎に生成しているのに対し，mod_mruby は複数のスクリプトで共有しているためだと考えられる．また，mruby スクリプトに Apache の内部処理を実装し，mod_mruby を介して mruby スクリプトをフックさせても，C 言語で実装した mod_hello と比較して，性能劣化は平均 12.1%程度であり，mod_lua の平均 50.5%の性能劣化と比較して，性能劣化が非常に少ない事が分かった．文字列を出力するだけの軽量な処理で，性能劣化が 12.1%程度あれば，動的コンテンツを扱う上ではほとんどボトルネックにならないと想定される．スクリプトの保守性や開発のし易さを考慮した場合，サーバ拡張のための一つの選択肢となり得るように考えられる．

```
require “Apache”

r = Apache::Request.new()

r.ap_auth_type = “Basic“
r.filename  = "/var/www/html/test.html"
r.user = “test“

Apache.rputs("the_request= "+r.the_request+"<BR>")
Apache.rputs("hostname= "+r.hostname+"<BR>")
Apache.rputs("status_line= "+r.status_line+"<BR>")
Apache.rputs("method= "+r.method+"<BR>")

Apache.return(Apache::OK)
```

図 7　request_rec 構造体の入出力

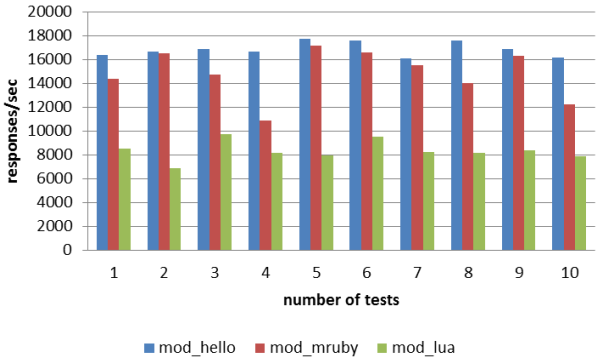

図 8　1 秒間のレスポンス数計測結果

表 1　テスト環境

クライアント	
CPU	Intel Core2Duo E8400 3.00GHz
Memory	4GB
NIC	Realtek RTL8111/8168B 1Gbps
OS	CentOS 5.6
サーバ	
CPU	Intel Xeon X5355 2.66GHz
Memory	8GB
NIC	Broadcom BCM5708 1Gbps
OS	CentOS 5.6
Middleware	Apache/2.2.3

5. まとめ

本稿では，Web サーバの機能拡張を支援するために，組み込みスクリプト言語である mruby に注目して，Web サーバの代表的なソフトウェアである Apache に，mruby スクリプトと Apache の内部処理を連携できるインターフェイス

を提供する事のできる Apache モジュール mod_mruby を開発した．これまで Apache の機能拡張は，C 言語で実装する必要があったが，Apache に関するドキュメントの少なさや保守性が低かったため，Apache モジュール開発の敷居が高かった．しかし，mod_mruby によって，mruby スクリプト上に Apache の内部処理を記述できる．また，mruby は Ruby の記述の仕方と同様の記述が可能であるため，Web に関わる技術者が容易に Apache の内部処理を拡張できるようになると考えている．

今後の課題として，まず機能面においては，Apache から mruby スクリプトをフックできるフェーズをさらに増やすための実装を追加する．次に，更に複雑な Apache の内部処理を扱うための mruby ライブラリを実装していく．さらに，Nginx でも mruby を組み込んでサーバ内部の処理を実装可能にするためのインターフェイスである ngx_mruby を開発予定である．その結果，複数の Web サーバソフトウェア上で Web サーバ用の mruby ライブラリを組み込む事で，Web サーバソフトウェア間の実装の違いを mruby ライブラリで吸収し，Web サーバ拡張機能を mruby での実装に統一できるのではないかと考えている．また，性能面においては，現状では状態遷移保存領域を複数のスクリプトで共有しているが，今後は，さらに高速に mruby スクリプトを動作させるようにするために，コンパイルされたバイトコードをキャッシュとして保存するためのキャッシュテーブルを実装する予定である．

謝辞 mod_mruby を実装するにあたり，協力していただいたファーストサーバ（株）津崎善晴氏に感謝する．

参考文献

1) Wikipedia, “ホスティングサーバ”, http://ja.wikipedia.org/wiki/%E3%83%9B%E3%82%B9%E3%83%86%E3%82%A3%E3%83%B3%E3%82%B0%E3%82%B5%E3%83%BC%E3%83%90
2) Wikipedia, “Web API”, http://en.wikipedia.org/wiki/Web_API.
3) 松本亮介，川原将司，松岡輝夫, “汎用性の高い大規模共有型 Web バーチャルホスティング基盤のセキュリティと運用技術の改善”，インターネットと運用技術シンポジウム 2011 論文集, 2011,31-38 (2011-11-24).
4) The Apache Software Foundation, "Apache HTTP SERVER PROJECT", http://httpd.apache.org/.
5) Roberto Ierusalimschy, Waldemar Celes and Luiz Henrique de Figueiredo , “The Programming Language Lua”, http://www.lua.org/.
6) Yukihiro Matsumoto, “mruby/mruby”, https://github.com/mruby/mruby.
7) nginx, “nginx”, http://nginx.org/ja/.