令和 2 年度

卒　業　論　文

Web アプリケーション開発に用いられる
主要スクリプト言語の実行時間に関する比較検討

HT18A102　山岡 風太

指導教員　久松 潤之 准教授

令和 2 年 12 月 14 日

大阪電気通信大学　総合情報学部　情報学科

令和 2 年度 卒業論文

Web アプリケーション開発に用いられる主要スクリプト言語の実行時間に関する比較検討

山岡 風太

内 容 梗 概

近年、スマートフォン等の個人用端末の普及によってインターネット上の WEB サービスは多くの人に利用されている。Web サービスを構築する際には PHP 等のスクリプト言語と呼ばれるプログラミング言語を用いるのが一般的である。スクリプト言語は C++ 等と比較するとコンパイルが不要という利点があるが、それによって実行時間が長くなりがちと言う問題が一般的に知られている。しかし、近年、需要の増大にともなって、各種スクリプト言語はバージョンアップ時に実行時間を含めたパフォーマンス問題の改善に力を入れている。また、Node.js など近年出現したスクリプト言語あるいはフレームワークは、初期段階からパフォーマンス問題に対して意欲的に取り組んでいる。そのため、これらのスクリプト言語のパフォーマンスは、従来考えられているよりも相当程度に改善されている可能性がある。そこで、本研究では各種スクリプト言語の最新のバージョンを対象にして、実行時間のパフォーマンスに関する検証を行う。

主 な 用 語

スクリプト言語　Ubuntu　Python [1]　PHP　Ruby　Node.js

目　　次

第 1 章　はじめに **1**

第 2 章　計測対象とするスクリプト言語の概要 **2**

2.1 Python 2

2.2 PHP 2

2.3 Ruby 2

2.4 Node.js 3

第 3 章　計測用プログラムの概要 **4**

3.1 バブルソート 4

3.2 フィボナッチ数列 4

3.3 ライプニッツ級数による円周率の算出 5

3.4 正規表現 6

第 4 章　計測環境 **10**

第 5 章　計測結果 **11**

5.1 バブルソート 11

5.2 フィボナッチ数列 11

5.3 ライプニッツ級数を用いた円周率の導出 11

5.4 正規表現 12

第 6 章　まとめと今後の課題 **16**

謝辞 **17**

参考文献 **18**

第1章　はじめに

Web サービスにおいては、Perl [2], Ruby [3], PHP [4] を始めとしたスクリプト言語と呼ばれるプログラミング言語を用いて開発するのが一般的となっている。これは、これらのプログラミング言語が C や C++ などに比べて、文字列操作や正規表現などの Web サービスの開発に必要となる機能を豊富にサポートしている事や、Web サービスの開発に特化したライブラリやフレームワークが数多く提供されている事などに起因する。スクリプト言語は Web 2.0 [5] のキーワードの下、多くの人々や企業が Web サービスを提供するようになるに伴って、急速に普及した。さらに iPhone や Android などのスマートフォンと呼ばれる携帯端末が普及すると、Web ブラウザ上に表示される Web ページがアプリケーションのように振る舞う、Web アプリケーションと呼ばれるものが注目を集めるようになった。これは、異なる OS に対してアプリケーションを提供する際に、Web ブラウザを利用する事によりプログラムを単一に保ち、開発コストを削減する効果が期待できる事などが理由に挙げられる。Web アプリケーションの開発に注目が集まるようになると、Go [6] や Web ブラウザ上での動作を前提としていた JavaScript をそれ以外の場面でも実行できるようにする Node.js [7] など新しいプログラミング言語やそれに関連する技術も多数登場した。

一般的に、スクリプト言語は C や C++ などのプログラミング言語に比べて実行速度が遅い。しかし、前述したように Web サービスや Web アプリケーションの開発が盛んになるにつれて、スクリプト言語開発者による最適化も盛んに行われるようになった。例えば、PHP は 2015 年に新バージョンとなる PHP7 [8] がリリースされたが、PHP7 は以前のバージョンである PHP5.6 と比較すると、ほとんどの互換性を維持したまま約二倍の性能向上に成功している。さらに命令呼び出し回数の削減や検索手法の改善、メモリ使用量の削減なども行われ、総合的なパフォーマンスが向上している。このような性能向上を伴うアップデートはほとんどのスクリプト言語においても同様に行われており、スクリプト言語の性能は従来考えられているよりも相当程度に改善されている可能性がある。

そこで本研究では、Web サービスや Web アプリケーションに用いられるスクリプト言語の最新バージョンを対象に、改めて各種パフォーマンスに関する検証を行う。具体的には、現在 Web サービスの開発に用いられる主要スクリプト言語である Python [1], PHP, Node.js, Ruby の最新バージョンを用いて、実行時間に関する比較検討を行う。

本論文の構成は以下の通りである。2 章では、実行時間の比較検討の対象となる 4 種類のスクリプト言語の概要を述べる。次に、3 章では、実行時間を比較検討するために本研究で作成したプログラムの概要を述べる。さらに、4 章では実行時間の比較検討に使用した計算機端末の構成と各種スクリプト言語のバージョンについて説明し、5 章では比較検討結果について述べる。最後に、6 章では、本論文のまとめと今後の課題を述べる。

第2章　計測対象とするスクリプト言語の概要

本章では、本研究において実行時間の比較検討の対象となる 4 種類のスクリプト言語の概要について述べる。

2.1 Python

Python は、オープンソースのオブジェクト指向スクリプト言語であり、1991 年に最初のバージョンが公開された。Python は、同じ処理を行うプログラムは誰が書いても同じになる事を目指して開発されたスクリプト言語であり、インデントによってプログラムのブロックを定義すると言う特徴がある。ほとんどのプログラミング言語においては、インデントは意味を持たず開発者によるプログラムの可読性を高めるために任意で使用されていたものであるが、Python ではインデントを用いてブロックを表現する事によってプログラムの構造が似たものとなるため、可読性が高いと言われている。Python は統計や解析、分析に長けたライブラリが充実しており、研究分野でよく使われる。特に近年では、ディープラーニングと呼ばれる機械学習手法に注目が集まっており、これを実現するためのライブラリが充実している Python には大きな注目が集まっている。

2.2 PHP

PHP は、オープンソースのスクリプト言語であり、1995 年に最初のバージョンが公開された。PHP は Hypertext Preprocessor の略称であり、主に動的な Web ページを生成する目的で開発が始まった。そのため、Web サービスや Web アプリケーション開発に関する標準ライブラリが豊富で、様々な Web サーバ上で利用されている。例えば、PHP で書かれた Web サービスには Facebook [9], Wikipedia [10], Slack [11] などが挙げられる。PHP の制御系は `<?php ?>` で囲まれた部分を読み取って解釈し、プログラムを実行する。ファイルの一部分に PHP のプログラムを記述できると言う性質上、マークアップ言語である HTML に埋め込んで利用される事も多い。また、C や Perl の影響を強く受けており、文法やプログラムの構造がこれらのプログラミング言語に類似しているため、それらのプログラミング経験者は学習が容易である。

PHP は 2015 年に新バージョンとなる PHP7 [8] がリリースされたが、PHP7 は以前のバージョンである PHP5.6 と比較すると、ほとんどの互換性を維持したまま約二倍の性能向上に成功している。さらに命令呼び出し回数の削減や検索手法の改善、メモリ使用量の削減なども行われ、総合的なパフォーマンスが向上している。

2.3 Ruby

Ruby は日本人によって開発されたオープンソースのオブジェクト指向プログラミング言語であり、1995 年に最初のバージョンが公開された。Enjoy Programming! を設計思想として開発されたプログラミング言語で、プログラムの記述量が少ない、構文がシンプル、標準ライブラリが高機能などの特徴がある。また、日本発のプログラミング言語では初めて国際標準規格に認定された。また、Ruby on Rails [12] と言う Web サービスや Web アプリケーションを開発するためのフレームワーク

が有名であり、このフレームワークを用いて開発された Web サービスや Web アプリケーションが世界中で数多く存在している。さらに、日本発のプログラミング言語であるため、日本語の資料が豊富であり、日本において初学者が最も容易に学習できるプログラミング言語の一つである。

2.4 Node.js

Node.js は非同期処理を行うアプリケーションを作成するために、これまで Web ブラウザ上で実行される事を前提としていた JavaScript をそれ以外の場面でも実行できるようにした JavaScript 実行環境であり、2009 年に最初のバージョンが公開された。メモリ消費量が少ないため、小規模の運用を行う場合では、他の環境と比べると総合的なパフォーマンスが高いとされる。また、ネットワーク通信やファイルの読み書きに関する処理において、処理待ちによってブロックされることが少ない非同期方式による処理を基本とするライブラリ設計がなされている。Web サーバでは、サーバへの接続台数が 1 万台を超えると処理が遅くなる C10K 問題 [13] と呼ばれる課題に悩まされてきたが、非同期処理を基本とした Node.js を用いる事で、この問題が比較的簡単に解決される。そのため、リアルタイム性の問われる Web サービスや Web アプリケーションの開発に長けているとされる。

第3章　計測用プログラムの概要

本章では、本研究において作成した4種類のスクリプト言語の実行時間を比較検討するためのプログラムについて述べる。本研究では、それぞれのスクリプト言語の基本的な実行時間を比較検討するためにバブルソート、フィボナッチ数列、ライプニッツ級数による円周率の算出プログラムを作成した。これに加えて、Web サービスや Web アプリケーションで頻繁に使用される正規表現の実行時間を比較検討するために、該当の機能を利用したプログラムも作成した。

3.1　バブルソート

まず始めに、数値配列のソートに関する実行時間を比較検討するために、バブルソートを行うプログラムを作成した。バブルソートは、ソートアルゴリズムの一つで実行の速度が遅く実用的ではないが、仕組みが理解しやすいため初学者の学習によく利用されている。このアルゴリズムは、ある数値配列における隣接する要素の大小比較および入れ替え操作を全ての要素に対して行う。一回目の比較が終わると、対象とする数列の最大値が確定し、該当の右端に移動した状態になる。二回目は、一回目の操作で確定した最大値を除いた数値配列に対して同様の操作を行う。この操作を繰り返す事で対象となる数値配列のソートを行う。

バブルソートプログラムの計算量は $\mathrm{O}(n^2)$ となる。これはソートアルゴリズムとしては計算量が多く、あまり実用的ではない。例えば、クイックソートの計算量は $\mathrm{O}(n \log n)$ であり、バブルソートよりも計算量が少ない。しかし、現在のコンピューターの性能では負荷の少ないクイックソートは、非常に短い時間で処理が終了してしまうため計測が難しいと考えた。そのため本研究では、計算量が多く、スクリプト言語の差が顕著に出ることが予想されるバブルソートを比較検討対象プログラムとする事にした。

本研究において、Python で実装したプログラムを図 3.1 に示す。このプログラムでは、テキストファイルから数値を読み取り、ソートした結果をテキストファイルに書き込む。この図の 1 行目で関数 bsort() を定義しており、2 行目から 6 行目でアルゴリズムを実装している。この図のその他の処理は、テキストファイルからソートする数値を読み込む処理である。

3.2　フィボナッチ数列

次に、スクリプト言語の再起呼び出しに関する実行時間を比較検討をするために、フィボナッチ数列を算出するプログラムを作成した。フィボナッチ数列とは「1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...」という様に、数列のそれぞれの値が 1 つ前と 2 つ前の値の和になる数列である。フィボナッチ数列は式 3.1 で表す事ができる。

$$\begin{aligned} F_0 &= 0 \\ F_1 &= 1 \\ F_{n+1} &= F_n + F_{n+1}(n \boxtimes 0) \end{aligned} \tag{3.1}$$

この漸化式を再起呼び出しを用いて、プログラム上に実装する。

本研究において、Python で実装したものを 図 3.2 に示す。この図の 3 行目に定義した fib() 関数を 7 行目で再起呼び出しを行っている。fib() の引数が 0 か 1 になるまで引数に指定される値を 1 ずつ減らしながら fib() を呼び出し続ける。また、PHP で実装したものを図 3.3 に示す。この図の 2 行目に定義した fib() 関数を 8 行目で再起呼び出しを行っている。二つのプログラムを比較すると言語ごとの細かい違いは存在しているが漸化式の再現方法は同じであることがわかる。他の Ruby , Node.js のプログラムについても、漸化式の実装方法は同じになるように注意して行った。

再起呼び出しを使ったプログラムは for 文などのループ処理に比べて、関数呼び出し時の負荷が何度も存在するため実行時間が増加する傾向にある。今回のプログラムも導出する数が 1 大きくなるごとに再起呼び出しの回数が指数関数的に増加する。今回作成したプログラムの計算量は $O((\frac{1+\sqrt{5}}{2})^{n-1})$ となる。このような時間のかかる漸化式を使った実装は避けられており実用的ではなく、一般的には、プログラミング言語の機能の一つであるメモ化などを使って実装することが多い。しかし、本研究では再起呼び出しに関する処理能力を計測することを目的としていたため、効率の悪い漸化式での実装を行った。

3.3 ライプニッツ級数による円周率の算出

本節では、スクリプト言語の実行速度を比較検討するための、円周率を求めるプログラムについて説明する。本研究では、円周率を求めるためにライプニッツ級数を用いた実装を行った。ライプニッツ級数は、式 3.2 で表せる。

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} = \frac{\pi}{4} \tag{3.2}$$

この式を展開すると式 3.3 のように表す事ができる。

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots = \frac{\pi}{4} \tag{3.3}$$

ライプニッツ級数においては、実行回数を増やすと導出される円周率の精度が向上する。例えば、10 億回実行すると円周率が 10 桁まで正しく導出される。ライプニッツ級数は円周率の計算方法の中では収束が遅く効率も悪いが、現在のコンピューターでその他の方法を使うと非常に早く収束してしまい、実行時間の計測が難しい。したがって、ライプニッツ級数はプログラミング言語の実行速度を計測するのに適していると予想される。

本研究において、Python で実装したものを図 3.4 に示す。この図の 6 行目から 9 行目で、式 3.2 の左辺にあたるものを実装している。また、10 行目で左辺を 4 倍することで円周率を求めている。次に、PHP で実装したものを図 3.5 に示す。この図の 2 行目から 11 行目で同様の実装をしており、その他の部分は導出した解をテキストファイルに書き込む処理である。この二つの図を比べると、ライプニッツ級数の実装方法にはほぼ差異がないことがわかる。本研究では、残りの 2 つのスクリプト言語での実装に関しても同様に、計算量が同じになるように注意して実装している。これらの

```python
def Sort(a):
    for i in range(len(a)):
        for j in range(len(a)-1, i, -1):
            if a[j] < a[j-1]:
                a[j], a[j-1] = a[j-1], a[j]
    return a

f=open('random_number.txt', 'r')
datalist=list(map(int,f.read().split()))
f.close
g = open('number-py.txt', 'w')
l=Sort(datalist)
for line in l:
    g.write(str(line)+" ")
g.close()
```

図 3.1　Python で実装されたバブルソートのプログラム

```python
import sys
n=int(sys.argv[1])
def fib(n):
    if n == 0 or n == 1:
        return n
    else :
        return fib(n-2)+fib(n-1)
fib(n)
```

図 3.2　Python で実装されたフィボナッチ数列を求めるプログラム

プログラムの計算量は O(n) となり、実行時間は実行回数に比例する。

3.4　正 規 表 現

最後に、スクリプト言語の正規表現に関する処理能力を比較検討するために、指定された文字列が URL かどうかを判別するプログラムを作成した。このプログラムでは、URL の混在した任意の行数の文字列が記載されているテキストファイルを開き、その内容から正規表現を用いて URL を抽出し、出力する。

本研究において、Python で実装したものを図 3.6 に示す。この図を見ると、正規表現を扱うライブラリである ”re” をインポートして関数 re.findall を使用している事が分かる。次に、PHP で実装したものを図 3.7 に示す。この図を見ると、PHP では preg-match 関数が用いられている事が分かる。。これらの関数は結果としては同じ振る舞いをするが、スクリプト言語ごとに実行速度が異なる結果になる事が予想される。

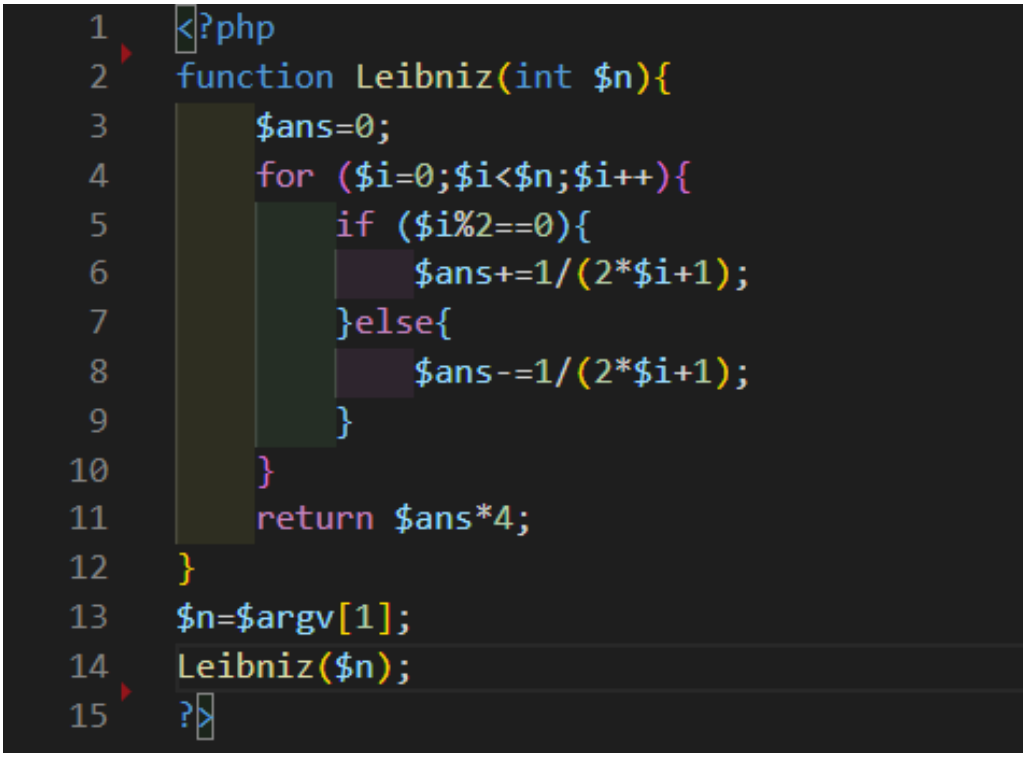
```python
import sys
n=int(sys.argv[1])
def fib(n):
    if n == 0 or n == 1:
        return n
    else :
        return fib(n-2)+fib(n-1)
fib(n)
```

図 3.3　PHP で実装されたフィボナッチ数列を求めるプログラム

```python
import sys

def Leibniz(n):
    ans=0
    for i in range(n):
        if i%2==0:
            ans+=1/(2*i+1)
        else:
            ans-=1/(2*i+1)
    ans=ans*4
    return ans
Leibniz(int(sys.argv[1]))
```

図 3.4　Python で実装されたライプニッツ級数を用いて円周率を導出するプログラム



```php
<?php
function Leibniz(int $n){
    $ans=0;
    for ($i=0;$i<$n;$i++){
        if ($i%2==0){
            $ans+=1/(2*$i+1);
        }else{
            $ans-=1/(2*$i+1);
        }
    }
    return $ans*4;
}
$n=$argv[1];
Leibniz($n);
?>
```

図 3.5　PHP で実装されたライプニッツ級数を用いて円周率を導出するプログラム

```python
import re
import sys
import fileinput

EXCUTION_TIMES=100000
def search(string):
    urlPattern='/(https?:\/\/(www\.)?[0-9a-z\-\.]+:?[0-9]{0,5})/'
    if re.findall(urlPattern, string):
        #print(re.findall(urlPattern, string))
        pass
for i in range(EXCUTION_TIMES):
    for line in fileinput.input():
        search(line)
```

図 3.6　Python で実装された正規表現を用いて URL を検出するプログラム

```php
<?php
$EXCUTION_TIMES=100000;
function search($str){
    if(preg_match('/(https?:\/\/(www\.)?[0-9a-z\-\.]+:?[0-9]{0,5})/', $str, $matches)){
        "echo $matches[0];";
    }
}
for ($i=0;$i<$EXCUTION_TIMES;$i++){
    $fp = fopen($argv[1], 'r');
    while (!feof($fp)) {
        $str = fgets($fp);
        search($str.'<br>');
    }
    fclose($fp);
}
?>
```

図 3.7　PHP で実装された正規表現を用いて URL を検出するプログラム

第4章　計 測 環 境

本研究では、3 章で述べたプログラムを実行し、スクリプト言語ごとに実行時間の比較検討を行うための計測環境について述べる。表 4.1 に、プログラムを実行する計算機端末の概要を示す。本研究では、この計算機端末にインストールされた Windows 上に存在する Windows Subsystem for Linux (WSL2) を用いて Linux の仮想環境を構築し、その環境下で作成したプログラムを実行する。Linux のディストリビューションは Ubuntu [14] とし、バージョンは 20.04. を用いた。また、本研究においてプログラムを実行する際に使用する各種スクリプト言語のバージョンを表 4.2 に示す。これらは、ともに 2020 年 8 月 29 日時点の最新バージョンである。

表 4.1　計測環境

CPU	Intel Core i5-8400 2.80GHz ×6
Memory	16GB
Storage	SSD 512GB
OS	Microsoft Windows 10 Home

表 4.2　スクリプト言語のバージョン

スクリプト言語	バージョン
Python	3.8.2
PHP	7.4.9
Ruby	2.7.1
Node.js	12.18.3

第5章　計 測 結 果

本章では、3 章で説明したプログラムを 4 章で述べた環境の下で実行した結果について述べる。尚、各種プログラムの実行時間は time コマンドを用いて計測した。

5.1　バブルソート

本節では、バブルソートを 4 種類のスクリプト言語で実装したプログラムを実行し、その結果を比較する。表 5.1 および表 5.2 に 100, 1000, 10000, 100000 個の要素に対してそれぞれのスクリプト言語で実装されたバブルソートを 20 回ずつ実行した時の実行時間の平均および分散を示す。

表 5.1 を見ると、4 種類のスクリプト言語で実装されたプログラムの内、最も実行時間が短いのは Node.js である事が分かる。特に、要素数が 100000 個の実行時間は他のスクリプト言語よりも非常に短くなっており、要素数が大きくなるにつれて Node.js を利用する事の利点が大きくなる。また、表 5.2 を見ると、どのスクリプト言語で実装されたプログラムも実行時間の分散は非常に小さく、実行時間の観点から見ると安定した動作をしている事が分かる。

5.2　フィボナッチ数列

本節では、フィボナッチ数を求めるプログラムを 4 種類のスクリプト言語で実装して実行し、その結果を比較する。表 5.3 および表 5.4 に、それぞれのスクリプト言語で実装されたプログラムを用いて 1, 5, 10, 15, 20, 25, 30, 35, 40 番目のフィボナッチ数を 20 回ずつ求めた時の実行時間の平均および分散を示す。

表 5.3 を見ると、20 番目までのフィボナッチ数までは Ruby で実装したプログラムの実行時間が最も短いが、求めるフィボナッチ数が増加するのに伴って、Node.js の実行時間が最も短くなる事が分かる。5.2 節から、バブルソートを実行するプログラムでも要素数が多い領域においては Node.js の実行時間の短さが際立っており、処理するデータが大きくなるにつれて Node.js の優位性が目立つ結果となっている。逆に、Python で実装したプログラムは求めるフィボナッチ数の数が小さい内は他のスクリプト言語と比較しても十分に実行時間が小さいが、求めるフィボナッチ数の数が増加するにつれて実行時間の伸び方が他のスクリプト言語を大きく上回り、40 番目のフィボナッチ数を求めた時の実行時間は最も大きくなる結果となった。

表 5.4 を見ると、5.1 節と同様に、どのスクリプト言語で実装されたプログラムも実行時間の分散は非常に小さく、実行時間の観点から見ると安定した動作をしている事が分かる。

5.3　ライプニッツ級数を用いた円周率の導出

本節では、ライプニッツ級数を用いた円周率の導出プログラムを 4 種類のスクリプト言語で実装して実行し、その結果を比較する。表 5.5 および表 5.6 に、それぞれのスクリプト言語で実装されたプログラムを用いて実行回数を 1 回から 1 億回まで変化させた時の実行時間の平均および分散を示す。尚、それぞれのプログラムは 20 回ずつ実行し、その平均および分散を用いた。

表 5.1　バブルソートの実行時間の平均

要素数	Python	PHP	Ruby	Node.js
100	0.018 sec	0.009 sec	0.276 sec	0.030 sec
1000	0.018 sec	0.010 sec	0.284 sec	0.030 sec
10000	0.076 sec	0.043 sec	0.345 sec	0.054 sec
100000	5.934 sec	3.148 sec	7.329 sec	0.030 sec

表 5.5 を見ると、ほとんどの場合において PHP を用いて実装したプログラムの実行時間が最も小さい事が分かる。5.1 および 5.2 節では、Node.js を用いて実装されたプログラムは処理するデータが大きな領域では特に有利になる結果となったが、ライプニッツ級数を用いた円周率の導出プログラムの場合、実行回数が 1000 万回を超えると実行時間の伸び方が、他のスクリプト言語と比較しても大きくなっている。これらの結果から、似たような処理を行うプログラムであっても、必ずしも実行時間の増加傾向は一致しない事が分かる。また、ライプニッツ級数を用いた円周率の導出プログラムでも Python で実装したプログラムは最も実行時間が大きくなっており、PHP で実装したプログラムの約 200 倍の実行時間を要している事が分かる。

表 5.6 を見ると、これまでと同様に、どのスクリプト言語で実装されたプログラムも実行時間の分散は非常に小さく、実行時間の観点から見ると安定した動作をしている事が分かる。

5.4 正 規 表 現

本節では、正規表現を用いた URL を検出するプログラムを 4 種類のスクリプト言語で実装して実行し、その結果を比較する。表 5.7 および表 5.8 に、それぞれのスクリプト言語で実装されたプログラムを用いて実行回数を 1 回から 1 万回まで変化させた時の実行時間の平均および分散を示す。尚、それぞれのプログラムは 20 回ずつ実行し、その平均および分散を用いた。

表 5.8 を見ると、Node.js と PHP で実装したプログラムの実行時間が非常に小さい事が分かる。また、Python で実装したプログラムの実行時間も十分に小さく、5.2 節や 5.3 節と比較すると、良い結果になっている事が分かる。一方で、Ruby で実装したプログラムの実行時間は、今回計測したものの中では実行時間がかなり大きくなる結果となった。

表 5.8 を見ると、これまでと同様に、どのスクリプト言語で実装されたプログラムも実行時間の分散は非常に小さく、実行時間の観点から見ると安定した動作をしている事が分かる。

これらの結果から、実行時間の観点では Node.js や PHP が有利になる傾向がある事が分かる。これは、[8] 等の公開情報からも分かるように、これらのスクリプト言語が他と比較しても様々なパフォーマンス問題に対して意欲的に取り組んでいるためであると推測される。しかし、スクリプト言語毎に得意または不得意な領域は存在するため、必ずしも全てのプログラムで同様の傾向が見られるとは限らない事も分かった。

表 5.2　バブルソートの実行時間の分散

要素数	Python	PHP	Ruby	Node.js
100	1.42×10^{-6}	0.00	2.52×10^{-5}	4.11×10^{-7}
1000	2.39×10^{-7}	2.61×10^{-7}	2.73×10^{-5}	3.24×10^{-5}
10000	8.92×10^{-7}	2.53×10^{-7}	2.28×10^{-5}	3.24×10^{-5}
100000	5.78×10^{-6}	7.87×10^{-7}	2.08×10^{-2}	4.27×10^{-6}

表 5.3　フィボナッチ数列の実行時間の平均

n 番目	Python	PHP	Ruby	Node.js
1	0.018 sec	0.274 sec	0.010 sec	0.038 sec
5	0.018 sec	0.272 sec	0.009 sec	0.028 sec
10	0.018 sec	0.270 sec	0.009 sec	0.028 sec
15	0.018 sec	0.274 sec	0.009 sec	0.028 sec
20	0.020 sec	0.274 sec	0.010 sec	0.031 sec
25	0.040 sec	0.280 sec	0.016 sec	0.030 sec
30	0.264 sec	0.371 sec	0.088 sec	0.040 sec
35	2.808 sec	1.302 sec	0.880 sec	0.145 sec
40	30.599 sec	11.947 sec	9.171 sec	0.149 sec

表 5.4　フィボナッチ数列の実行時間の分散

n 番目	Python	PHP	Ruby	Node.js
1	8.32×10^{-7}	2.87×10^{-5}	3.66×10^{-7}	1.92E-03
5	3.58×10^{-7}	4.49×10^{-5}	1.97×10^{-7}	3.45×10^{-7}
10	2.39×10^{-7}	3.41×10^{-4}	1.97×10^{-7}	2.74×10^{-7}
15	1.55×10^{-7}	4.00×10^{-4}	2.61×10^{-7}	2.39×10^{-7}
20	3.03×10^{-7}	3.01×10^{-4}	1.34×10^{-7}	2.80×10^{-5}
25	2.53×10^{-7}	5.71×10^{-5}	2.21×10^{-7}	3.79×10^{-7}
30	4.22×10^{-6}	1.04×10^{-4}	1.04×10^{-6}	6.42×10^{-7}
35	1.64×10^{-3}	8.70×10^{-4}	4.02×10^{-4}	1.19×10^{-6}
40	1.96×10^{-1}	1.79×10^{-1}	4.20×10^{-2}	2.83×10^{-6}

表 5.5　ライプニッツ級数を用いた円周率の算出の実行時間の平均

実行回数	Python	PHP	Ruby	Node.js
1	0.018 sec	0.010 sec	0.294 sec	0.028 sec
10	0.018 sec	0.010 sec	0.288 sec	0.028 sec
100	0.018 sec	0.028 sec	0.292 sec	0.028 sec
1000	0.018 sec	0.010 sec	0.297 sec	0.028 sec
10000	0.019 sec	0.011 sec	0.300 sec	0.033 sec
100000	0.031 sec	0.011 sec	0.304 sec	0.035 sec
1000000	0.151 sec	0.015 sec	0.304 sec	0.058 sec
10000000	1.399 sec	0.058 sec	0.437 sec	0.804 sec
100000000	13.646 sec	0.058 sec	1.691 sec	8.300 sec

表 5.6　ライプニッツ級数を用いた円周率の算出の実行時間の分散

実行回数	Python	PHP	Ruby	Node.js
1	8.32×10^{-7}	3.45×10^{-7}	6.60×10^{-5}	5.68×10^{-7}
10	8.32×10^{-7}	2.53×10^{-7}	8.30×10^{-5}	1.97×10^{-7}
100	5.16×10^{-7}	1.68×10^{-7}	7.74×10^{-5}	1.68×10^{-7}
1000	5.16×10^{-7}	4.71×10^{-7}	6.34×10^{-4}	1.68×10^{-7}
10000	2.53×10^{-7}	4.74×10^{-7}	9.40×10^{-4}	1.67×10^{-6}
100000	3.03×10^{-7}	9.37×10^{-7}	2.76×10^{-4}	8.00×10^{-7}
1000000	1.63×10^{-6}	8.00×10^{-7}	2.76×10^{-4}	1.61×10^{-6}
10000000	2.19×10^{-3}	2.17×10^{-6}	2.73×10^{-4}	1.45×10^{-3}
100000000	3.99×10^{-2}	2.17×10^{-6}	1.94×10^{-3}	2.16×10^{-2}

表 5.7　正規表現の実行時間の平均

実行回数	Python	PHP	Ruby	Node.js
1	0.021 sec	0.009 sec	0.269 sec	0.038 sec
10	0.022 sec	0.010 sec	0.292 sec	0.054 sec
100	0.043 sec	0.014 sec	0.490 sec	0.054 sec
1000	0.243 sec	0.062 sec	2.603 sec	0.114 sec
10000	2.370 sec	0.532 sec	23.386 sec	0.564 sec

表 5.8　正規表現の実行時間の分散

実行回数	Python	PHP	Ruby	Node.js
1	1.42×10^{-6}	0.00	2.52×10^{-5}	4.11×10^{-7}
10	2.39×10^{-7}	2.61×10^{-7}	2.73×10^{-5}	3.24×10^{-5}
100	8.92×10^{-7}	2.53×10^{-7}	2.28×10^{-5}	3.24×10^{-5}
1000	5.78×10^{-6}	7.87×10^{-7}	2.08×10^{-2}	4.27×10^{-6}
10000	4.25×10^{-3}	7.92×10^{-5}	5.88×10^{-1}	7.09×10^{-5}

第6章　まとめと今後の課題

本研究では、主要な 4 種類のスクリプト言語である Python, PHP, Ruby, Node.js のパフォーマンスの検証を実行速度の観点から行った。また、その結果を表にして示しその考察を行った。

今後の課題としては、パフォーマンス検証の評価を実行速度だけではなく、CPU の使用率やメモリー使用率などを加味して行うべきだろう。

謝　　辞

本研究と本論文を終えるにあたり、御指導、御教授を頂いた久松潤之准教授に深く感謝致します。また、学生生活を通じて、基礎的な学問、学問に取り組む姿勢を御教授頂いた、登尾啓史教授、升谷保博教授、渡邊郁教授、南角茂樹教授、鴻巣敏之教授、北嶋暁教授、大西克彦教授に深く感謝致します。

本研究期間中、本研究に対する貴重な御意見、御協力を頂きました久松研究室の皆様に心から御礼申し上げます。

参 考 文 献

[1] Python. available at https://www.python.org/.

[2] The Perl Programming Language. available at https://www.perl.org/.

[3] Ruby Programming Language. available at https://www.ruby-lang.org/.

[4] PHP: Hypertext Preprocessor. available at https://www.php.net/.

[5] Tim O'Reilly. What is Web 2.0. Technical report, O'Reilly, 2005. available at https://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html.

[6] The Go Programming Language. available at https://golang.org/.

[7] Node.js. available at https://nodejs.org/.

[8] PHP 7.0.0 release announcement. available at https://www.php.net/releases/7_0_0.php.

[9] Facebook. available at https://www.facebook.com/.

[10] Wikipedia. available at https://www.wikipedia.org/.

[11] Slack. available at https://slack.com/.

[12] Ruby on Rails. available at https://rubyonrails.org/.

[13] Dan Kegel. The C10K problem, 1999. available at http://www.kegel.com/c10k.html.

[14] Ubuntu: Enterprise Open Source and Linux. available at https://ubuntu.com/.