2E-06

スクリプト系言語の並列実行記法と実行速度の比較

黒瀬 浩†

金沢工業大学工学部†

1. はじめに

近年，多くのプログラミング言語が発表されている．各言語とも利用環境，ソフトウェア実行上の問題，入門者にも理解しやすいソースコード記法が工夫されている．本稿では，スクリプト言語の並行実行の記法と実行速度を 3 つの言語で比較し，初学者にとって並行実行処理の学習しやすさや実行速度について考察する．

2. 比較対象

本節では並列処理の前提と比較するプログラミング言語について述べる．

2.1 並列処理

並列処理では，同期，排他制御，メッセージ交換などの機能が必要であるが，比較の単純化のため，まず同一処理を与える値を変えて並列に動作させる．プログラミング言語で管理する並行実行リソースの空き状況の管理はプログラミング言語で管理する．並行で動作する関数を MAP 処理により並列実行する．関数は特定の処理を実施し戻り値を返す．戻り値を集め終えたら FILTER 処理により不要な戻り値を除去する．並行実行数の制御はプログラミング言語の API に依存する．

2.2 プログラミング言語

初学者でも学習しやすいスプリプト系言語から特徴の異なる 3 つの言語を選定した．

1) Python3
   MAP,FILTER 関数よりリスト内包が推奨されている．並行実行はいくつかのパッケージが用意されているが joblib[1]を用いる．
2) Raku(Perl6)
   豊富な演算子，メソッドが標準で使用可能で標準機能として実行順を制御する場合としない場合の比較を行う．
3) Elixir
   Erlang 言語の VM,パッケージを利用する関数型プログラミング言語で，並列処理は標準で実装されている．基本的な Task パッケージを用いる．

Comparison of parallel execution notation and execution speed of scripting programming languages
† KUROSE Hiroshi, Kanazawa Institute of Technology

3. 並列実行記法

比較対象のプログラミング言語の特徴的な部分と並列実行制御可能事項について以下に示す．ここで，func は並行動作する関数である．

3.1 Python3

Python ではリスト内包により後ろで生成した数列を前の delayed に実行する関数を高階関数として渡し，Parallel で並行動作させている．実行する順序と渡される引数を把握するには python のリスト内包処理の理解が必要である．

```
res = Parallel(n_jobs=-1)\
        ( [delayed(func)(n) for n
                    in range(2,100)] )
```

ここで n_jobs は並列実行数で，-1 を指定するとシステムのプロセッサ数に合わせて実行される．Parallel の引数ではスレッドかプロセスかを指定する backend，タイムアウト値，1 プロセスに割り当てるタスク数を指定する batch_size,事前に生成するタスク数を指定する pre_dispatch などのパラメータが利用できる．

3.2 Raku(Perl6)

標準で用意されている hyper[2]と race[3]メソッドが利用できる．メソッドチェーンによりデータの生成から順に処理を記述できるので理解しやすい．

```
(
  [2..^100]
  .hyper(batch=>64, degree=>4)
  .map( { func($_) } )
).list
.say
```

ここで hyper を race にすると実行順制御を入力順から変更できる．これらのメソッドでは batch はバッチサイズ，degree は並行実行数を指定する．

3.3 Elixir

Raku と同様にパイプ演算子|>によりデータの発生から処理を順に記載できる．Task.async で並列実行し Task.await で終了を待ち合わせす

る．実行順通りに並べ直す場合は，Task の ID を戻り値に含め，ソートすれば良い．

```
2..(100-1)
|> Enum.map(
     &Task.async(Func, :func, [&1]) )
|> Enum.map( &Task.await( &1 ) )
|> IO.inspect
```

4. 並列動作する関数の記法の比較

実行時間を評価するために実行速度が引数により異なる例題として引数が素数であれば引数を，素数でなければ無効な値を返すプログラムを考える．各言語で特徴的な部分を以下に示す．

4.1 Python

python では if，while などの制御構造を用いた手続き型の処理となる．

```
def prime(i):
    if i==2: return i
    elif i%2==0: return None
    j=3
    while j<i:
        if i%j==0: return None
        j+=2
    return i
```

4.2 Raku(Perl6)

Raku では，同様の処理を行うのに複数の記法があるが，ここでは，関数の多重定義と while 制御構文を用いた．

```
multi sub prim($i where $i==2){ 2 }
multi sub prim($i where $i%%2){False}
multi sub prim($i){
  my $j=3;
  while ($j < $i) {
      if  $i %% $j { return False }
      $j+=2;
  }
  $i;
}
```

4.3 Elixir

Elixir では引数のマッチングによる関数多重定義と，再帰呼び出しによる繰り返し処理を記載できる．

```
def mods(x, n) do
  cond do
    x==n          -> n
    rem(n,x)==0   -> Nil
    x>n           -> Nil
    true          -> mods(x+2,n)
  end
end
def prime(2), do: 2
def prime(4), do: Nil
def prime(n), do: mods(3, n)
```

5. 性能評価

正数 10^m までの素数リストの取得に要する時間を測定する．動作環境を表 1 に，結果を表 2 に示す．

表 1 測定環境

項目	諸元
CPU	Intel Core i7 2.3GHz 6 コア
メモリ	64GB
OS	macOS 10.15.2
Python	3.7.5 Anaconda
Raku	Rakudo Star 2019.03.1 MoarVM
Elixir	1.9.1 Erlang/OTP 22

表 2 実行結果(m=5)

言語	パラメータ	時間[s]
Python	njob=1	20.345
	njob=2	9.481
	njob=4	5.382
	njob=8	4.328
	njob=12	4.519
	threading, njob=1	20.237
	threading, njob=2	21.545
	threading, njob=4	20.792
	threading, njob=8	20.582
	threading, njob=12	20.432
Raku	hyper, degree=1	66.721
	hyper, degree=2	36.900
	hyper, degree=4	23.153
	hyper, degree=8	15.586
	hyper, degree=12	14.877
	race, degree=1	67.121
	race, degree=2	38.370
	race, degree=4	20.383
	race, degree=8	15.470
	race, degree=12	13.569
Elixir	標準設定から変更なし	0.956

6. おわりに

ソースコードの可読性では，Raku や Elixir が Python より勝り，実行時間では，Elixir，Python，Raku の順であることがわかった．Python の joblib では，スレッドよりプロセスの方が高速である．今後は，排他制御，メッセージ交換について比較したい．

参考文献

[1] 'Joblib: running Python functions as pipeline jobs,' https://joblib.readthedocs.io/en/latest/

[2] 'method hyper,' Raku documentation, https://docs.raku.org/routine/hyper