Vol.2018-EMB-48 No.5
2018/6/29

関数型言語ElixirのIoTシステムへの導入に向けた基礎評価

高瀬 英希[1,a)]　上野 嘉大[2]　山崎 進[3]

概要：2012 年に登場した Elixir は，Erlang VM 上で動作する関数型言語であり，分散システム対応でスケールしやすく，軽量で耐障害性が高いという特徴がある．習熟容易性および開発生産性が高く，並列処理のプログラミングが容易に実現できる．Elixir の Web アプリケーションフレームワークとしては Phoenix が知られている．本稿では，IoT システムの構築における Elixir の有用性を議論する．IoT システムにおいて多く採用される IoT ボードに Elixir/Phoenix の実行環境を整備し，種々の性能を評価できる動作するベンチマークスイートを用意する．これによって各種 IoT ボードにおける Elixir/Phoenix の実行性能を基礎評価し，IoT システムへの導入に向けた有用性を定量的に議論する．評価の結果，Elixir の並列処理性能は IoT システムにおいても発揮されること，Phoenix サーバについても IoT システムに十分な応答性能を得られることが示された．

An Empirical Evaluation to Performance of Elixir for Introducing IoT Systems

Hideki Takase[1,a)]　Yoshihiro Ueno[2]　Susumu Yamazaki[3]

1. はじめに

IoT（Internet of Things）の普及は，我々の生活に多様なデバイスと大量のデータをもたらしている．ネットワークインフラの整備に後押しされ，あらゆるモノやコトさらにはヒトまでインターネットに接続して情報処理される時代が訪れている．IoT のシステムアーキテクチャを構成する要素は，外界環境の直近に位置してセンシングならびにアクチュエーションを担うデバイス，デバイスの近傍に位置して収集したデータを加工しつつクラウドサーバに中継するエッジサーバ，および，収集されたデータを統計解析するクラウドから構成される．IoT システムでは，それぞれの構成要素が持つ計算機特性を集結させることで，社会的価値を創造していくことが期待されている [1]．

IoT システムのデバイスおよびエッジサーバについては，限られた計算資源のもとで要求性能を発揮しつつ省電力化を達成することが求められる．また，末端であるデバイスの個数の増大に伴い，データ通信量の爆発的増加に応えうるネットワークトラフィックの最小化が課題となる．ただし，現状の IoT システム開発では，各構成要素の開発が個別に行われており，構成レイヤ間の開発手法の相違によって開発生産性の低下が生じることがある．

特に組込み技術が活用されるデバイスおよびエッジにおいては，IoT システムの構築に向けた開発生産性の向上が不可欠である．この解決に向けたひとつの方策は，抽象度の高いプログラミング言語を導入することが挙げられる．組込み分野では旧来より C/C++が採用されることが多く，これは低レイヤのプログラミングや性能要件が厳しい機能の実現には役立つが，習熟容易度や開発生産性については難がある．例えば Python や Ruby などのスクリプト言語は，少ないコード行数で所望の機能を実装することに貢献する．これらのプログラミング言語を組込み機器に応用するための開発フレームワークとしては，micro-python[2] や mruby[3] の研究開発が進んでいる．

我々は，IoT 分野における Elixir[4], [5] の可能性に着目する．Elixir は 2012 年に登場した関数型言語であり，処理の振る舞いではなくデータの扱いを直接的に操作するため

1　京都大学
2　有限会社デライトシステムズ
3　北九州市立大学
a)　takase@i.kyoto-u.ac.jp

のライブラリや記法が豊富に整備されている．習熟容易性および開発生産性に優れており，並列処理のプログラミングが容易に実現できるという利点がある．また，Elixir は軽量かつ頑強なプロセスモデルを採用しており，分散処理に向いていてシステムのスケーラビリティを得やすいという特徴を持つ．Elixir の Web フレームワークとしては，同時接続性と応答性に優れる Phoenix が知られている．これらの特徴は，IoT システム開発における生産性の向上に対して大きな利点になりえると考える．

本稿では，IoT システム開発への Elixir/Phoenix の導入に向けた足がかりとして，IoT ボード上におけるこの基礎性能を評価する．Linux が動作する種々の IoT ボード上に Elixir/Phoenix の動作環境を整備し，様々な性能を評価するためのベンチマークアプリケーションを用意する．ベンチマーク評価によって得られた基礎性能より，IoT システム開発における Elixir の有用性を定量的に議論する．

本稿の構成は，以下の通りである．2 章では，本研究の対象である関数型言語 Elixir について解説する．3 章では，評価環境として整備した IoT ボードとそれぞれの仕様を示す．4 章では，基礎評価のために用意したベンチマークアプリケーション，および，評価用のに整備したツールキットを紹介する．5 章では，基礎評価の結果を示し，IoT システムへの導入に向けた Elixir の可能性を考察する．最後に 6 章では，本稿のまとめと今後の展望を述べる．

2. Elixir

Elixir は，2013 年に登場した新しいプログラミング言語である．Elixir は次のような特徴を持つ．

(1) Elixir は，従来の関数型言語に比べて言語仕様や概念がシンプルであるが応用が効くようにデザインされている．たとえば，パイプライン演算子 |>と MapReduce モデル [6] に基づく Flow[7] を用いて**図 1** のようにデータを加工しながら計算が進行する．このプログラムでは，1 から 1000 までのリストを生成し，各コアにリストの要素を分配した後，各要素について関数 foo と bar を順番に適用してから結果をリストに戻す．このように，習得が容易でありかつ記述性に優れているため，結果的として生産性が向上する．

(2) Elixir は関数型言語であり，全ての変数はイミュータブルである．通常の言語ではマルチコアによる並列処理を行なってもコア間の動機や排他制御などが多々発生するため，コア数の増加に対し実行速度の向上が鈍化する傾向がある．しかし，Elixir ではイミュータブル特性によりマルチコアによる並列処理を行うとコア数に比例して実行速度が向上する [8]．たとえば，図 1 のプログラムコードを記述すると，各コアにリスト要素を分配して実行される．

(3) Elixir で実装されたウェブフレームワーク Phoenix[9] は，Ruby で実装された同様のフレームワークである Ruby on Rails と同等以上の生産性を誇りながら，レスポンス性が極めて高い．[10] によれば，Phoenix によって構築されたサーバは Ruby on Rails のそれよりも 10.63 倍のスループットを達成できたという評価結果が報告されている．このため，データベースをもとに複雑な計算をしてグラフィカルな表示をするようなウェブサイトを構築したときに，大量のアクセスに応えることができる．このことにより，IoT のバックエンドサーバの構築に Phoenix を用いることで，極めてレスポンス性の高いシステムを構築できる．[8] では，Elixir/Phoenix の IoT サーバにおける性能向上の貢献が示されている．加えて，もしデバイス側でも Phoenix を活用することができれば，リアルタイム性に優れた IoT システムを構築できる潜在能力がある．

(4) 1 つ 1 つのタスク（プロセス・タスク）ごとにガーベッジコレクションを含む堅牢なメモリ管理されている．加えてレスポンス性が高く，再起動を高速に行える．そのため，Elixir や Phoenix では，try/catch のような例外処理をするのではなく，障害が起こったら再起動し，外部で動作している障害監視プロセスで例外処理を行う．これにより，例外処理記述がとてもシンプルになり，耐障害性の高いシステムを構築できる．この性質は IoT バックエンドサーバーで有用である．もしデバイス側でも用いることができれば，耐障害性が高く，かつ例外処理を容易に記述できる可能性が高い．

```
1..1000
  |> Flow.from_enumerable()
  |> Flow.map(foo)
  |> Flow.map(bar)
  |> Enum.to_list
```

図 1 Flow を使ったリスト操作の Elixir 記述
Fig. 1 Elixir code of list manipulation using Flow.

3. IoT ボード

本章では，Elixir の基礎性能の評価のために用意した IoT ボードとそれぞれの仕様を示す．本研究において選定した IoT ボードの要件としては，Linux が動作して容易にインターネット接続環境が容易に整備できるものとした．

3.1 Raspberry Pi 3 Model B

Raspberry Pi は，IoT 分野で最も脚光を浴びているシングルボードコンピュータである．もともとは教育用途に開発されていたものの，低価格でありながら十分な性能のプロセッサコアが搭載されており，その扱いやすさによって IoT システムの実製品にも採用されることがある．

本研究では 64-bit のプロセッサコアが搭載されている Raspberry Pi 3 Model B（以降，RP3B）[11] を用いた．主

な性能仕様は下記のとおりである．

- 1.2GHz Cortex-A53 (ARMv8 64-bit, 2.3DMIPS/MHz), 4-core processor
- 1GB LPDDR2 memory
- 150Mbps 2.4GHz 802.11b/g/n WiFi

Linux カーネルとしては，Debian ベースの純正 OS である Raspbian Stretch with Desktop 4.9[12] および Ubuntu MATE 16.04[13] の 2 種類を用いた．インターネット接続はオンボードの BCM43143 によって無線化した．

3.2 ODROID-XU3

ODROID は，Hardkernel 社が販売しているシングルボードコンピュータである．主に携帯型情報通信機器向けの SoC を搭載しており，OS として Android や Ubuntu などが配布されているため，スマホやタブレットなどの開発における評価ボードとして活用される．

本研究では Samsung 社 Exynos 5 Octa 5422[14] を搭載した ODROID-XU3[15] を用いた．Exynos 5 Octa 5422 には，計 8 個の物理コアが集積されている．arm 社の big.LITTLE アーキテクチャを採用しており，高性能なコアおよび電力効率に優れたコアとの動作コアの切替えや動的電圧周波数制御（DVFS）の適用が可能である．これによって，アプリケーションの振る舞いに応じた省電力タスクスケジューリングを実現することができる．ODROID-XU3 の主な性能仕様は下記のとおりである．

- 2.0GHz Cortex-A15 (ARMv7 32-bit, 3.5DMIPS/MHz) 4-core & 1.4GHz Cortex-A7 (ARMv7 32-bit, 1.9DMIPS/MHz) 4-core processor
- 2GB LPDDR3 memory
- 300Mbps 2.4GHz 802.11n/g/b WiFi

Linux カーネルには HardKernel 社が公式配布している ubuntu-16.04-mate-odroid-xu3-20170731[16] を用いた．なお，ODROID-XU3 は WLI-UC-G301N USB-WiFi adaptor によって無線接続化することとした．

3.3 ZYBO

FPGA の主要ベンダである Xilinx 社は，次世代の SoC プラットフォームとしてハードマクロのプロセッサおよび FPGA を 1 チップに集積した Zynq-7000 SoC シリーズを発表している．柔軟性の求められる逐次処理はソフトウェアで，応答性の求められる並列処理はハードウェアでそれぞれ実装することで，高い処理性能と柔軟性を兼ね備えた組込みシステムを実現できることが期待される．

本研究では XC7Z010-1CLG400C を搭載した ZYBO[17] を用いた．PS 部の主な性能仕様は下記のとおりである．

- 650MHz Cortex-8 (ARMv7 32-bit, 2.0DMIPS/MHz) 2-core processor

```
defmodule LeibnizFormula do
  def calc(n) do
    s = 0..n
      |> Enum.reduce( 0, fn (x, acc) ->
        acc + (:math.pow(-1, x) / (2*x + 1)) end)
    s*4
  end
end
```

図 2 `leibniz` の Elixir 記述
Fig. 2 Elixir code of `leibniz`.

- 512MB DDR3 memory
- Trimode (1Gbit/100Mbit/10Mbit) Ethernet PHY

Linux カーネルには Xillybus 社が公式配布している Xillinux-2.0[18] を用いた．ZYBO はボード上の Ethernet コネクタにより有線接続することとした．

なお，本研究では活用しないが，ZYBO の SoC には Cortex-A9 dual core に加えて 28k LUTs および 240KB BRAM の FPGA が集積されている．

4. ベンチマークアプリケーション

ベンチマークとは，コンピュータシステムのハードウェアやソフトウェアの性能を測定する指標のことを指し，評価対象において相対的な性能を表す指標として用いられる．異なる SoC 構成やアーキテクチャを持つシステム間において，定量的な性能比較の手段を提供する．ベンチマークアプリケーションの整備の際に重要なことは，対象システムを比較評価するための項目である．本章では，Elixir の基礎性能の評価のために用意したベンチマークアプリケーションと，それぞれの評価項目を示す．

4.1 ライプニッツ級数

小数点演算の性能の評価を目的としてライプニッツ級数を用いる．ライプニッツ級数は下記の式によって円周率の値を計算する．

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

単純な数式ではあるが，和の上限が大きい場合に収束が遅くなるため性能評価に適している．

本アプリケーション `leibniz` の Elixir 記述を**図 2** に示す．和の上限は引数で指定するようにしている．

4.2 フィボナッチ数列

整数演算およびスタック使用時の性能を評価するため，フィボナッチ数列を用いた．

$$F_0 = 0, F_1 = 1, \qquad F_{n+2} = Fn + Fn + 1 \tag{2}$$

本アプリケーション `fib_simple` の Elixir 記述を**図 3** に示す．漸化式の項数は引数で指定するようにしている．このように Elixir は再帰計算を容易に記述することができる．

```
defmodule FibonacciSimple do
  def fib(n), do: fib_iter(0, 1, n, 1)

  defp fib_iter(_, _, n, i)
    when i > n, do: []
  defp fib_iter(a, b, n, i),
    do: [a | fib_iter(b, a + b, n, i + 1)]

end
```

図 3 fib_simple の Elixir 記述
Fig. 3 Elixir code of fib_simple.

4.3 複数プロセスでのフィボナッチ数列

Elixir の大きな利点のひとつである並列処理性能を評価するため，複数プロセスでフィボナッチ数列を求めるベンチマークアプリケーション fib_mulit を用意した．ソースコードは書籍 [4] の Section 14.5 で示されており，サポートページ [19] で公開されているものを用いた．フィボナッチ数の計算を担うプロセスと，空きの計算プロセスに対して仕事を割り振るスケジューラプロセスで構成され，プロセス間のメッセージ交換によって複数のフィボナッチ数列を並列に処理していく．引数としては，各数列の項数，同時処理する数列の個数，および，並列実行の計算プロセス数を与える．

4.4 大規模データのストリーム処理

大規模データの処理時に掛かる性能を評価するため，巨大な CSV ファイルをストリーム処理するアプリケーション [20] を用意した．本アプリケーション elixir_agg_csv の Elixir 記述を図 4 に示す．CSV 形式で記録されたファイルデータをストリームで受け付け，前処理したのちに 2 列目のデータの出現数を集計して多い順でソートする．Elixir における代表的な並列プログラミングのためのライブラリである Flow を用いており，より実用的なアプリケーションとなっている．

4.5 ロジスティック写像

Elixir の動作する Erlang VM には，他言語で実装されたモジュールを呼び出すことができる NIF（Native Implemented Functions）[21] という機能が提供されている．本機能の有効性を評価するため，メモリ安全なネイティブコンパイラ言語である Rust のモジュールを NIF 経由で呼び出すライブラリである Rustler[22] を用いる．

NIF 機能の評価のために，生物の個体数の時系列に対する変遷のモデルであるロジスティック写像 [23] を用意した．ロジスティック写像は単純な式でありながら解が複雑な振る舞いをすることが知られている．本研究では整数演算とするために次の漸化式を用いた．

$$X_{i+1} = \mu_p X_i (X_i + 1) \bmod p \tag{3}$$

```
defmodule ElixirFlow do
  def run( filename ) do
    result =
      filename
      |> File.stream!
      |> Flow.from_enumerable()
      |> Flow.map( &( String.replace(&1, ",", "\t")
        |> String.replace("\r\n", "\n")
        |> String.replace("\"", "")))
      |> Flow.map(&(&1 |> String.split("\t")))
      |> Flow.map(&Enum.at(&1, 2 - 1))
      |> Flow.partition
      |> Flow.reduce(
        fn -> {} end, fn(name, acc)
        -> Map.update(acc, name, 1, &(&1 + 1)) end)
      |> Enum.sort(&(elem(&1, 1) > elem(&2, 1)))
    IO.inspect(result)
  end
end
```

図 4 elixir_agg_csv の Elixir 記述
Fig. 4 Elixir code of elixir_agg_csv.

NIF 機能のベンチマークアプリケーションとして用意した logistic_map[24] は，Rustler の使用が選択でき，双方の比較評価が可能である．また，Flow による並列プログラミングで記述されており，並列化の段数も指定できる．

4.6 Phoenix サーバの接続負荷

Elixir の Web フレームワークである Phoenix について，サーバのレイテンシとスループットの性能を評価する phoenix-showdown[10], [25] を用意した．本アプリケーションでは，URL の:title 部分をコントローラに渡し，モデルの返り値を想定した固定値の配列をビューに渡す．ビューでは受け取った配列から HTML を生成して Web クライアントに返す．

Phoenix は PostgreSQL や MySQL などの DB とあわせて使用されることが多いが，phoenix-showdown は DB を使用しないため性能評価に扱いやすい．つまり，本アプリケーションは DB アクセスの影響を除外して Phoenix サーバの性能を測定できる．

5. 基礎評価

本章では，ここまでで用意した IoT ボード上におけるベンチマークアプリケーションの評価結果を示し，IoT システムにおける Elixir の性能を定量的に議論する．

5.1 評価方法と比較指標

Elixir のバージョンについては，公平性を期するために各 IoT ボード上で Elixir 1.6.4 (compiled with OTP 20) をソースビルドして統一した．Erlang/OTP のビルド時には HiPE（High Performance Erlang）[26] を導入している．

ライプニッツ級数，フィボナッチ数列および大規模デー

タ処理のアプリケーションについては，それぞれの実行時間を:timer.tc 関数によってマイクロ秒単位で計測した．Phoenix サーバについては，ネットワーク越しのラップトップから wrk コマンドを用いて複数スレッドで同時接続を試みることで，通信のレイテンシやスループットの性能を評価した．

ここで，3 章で整備した IoT ボードについてそれぞれの比較評価で得られる指標を整理する．

- 各 IoT ボードに搭載されるプロセッサコアの動作周波数やマイクロアーキテクチャの相違を議論できる．RP3B と ODROID-XU3 および ZYBO では，ARM プロセッサコアの命令長および命令セットも異なる．
- RP3B は Raspbian と Ubuntu の 2 種類を用意した．これらの比較によって OS ディストリビューション間の相違を議論できる．なお，Ubuntu のバージョンは全ての IoT ボードで 16.04 LTS である．
- インターネット接続の形態の相違を議論できる．RP3B および ODROID-XU3 は WiFi で，コア性能の劣る ZYBO は Ethernet で接続している．

5.2 基礎評価の結果

表 1 にライプニッツ級数計算およびフィボナッチ数列計算のパラメータを固定して評価した実行時間を示す．leibniz の級数は 10^8，fib_simple の数列は 50_000，fib_multi は 37 までの数列を 6 個同時に 10 プロセス並列で計算するパラメータ設定としている．

まず注目すべきは，やはり並列性能である．fib_multi では，物理コアを 8 個持つ ODROID-XU3 が最良の性能を発揮した．leibniz は単一プロセスの処理であるため ODROID-XU3 ではマスタコアの 1.4GHz Cortex-A7 (32-bit) で動作するが，1.2GHz Cortex-A53 (64-bit) の RP3B を 2 倍以上も凌駕する結果が得られた．対して fib_simple では RP3B/Ubuntu の実行時間が最小となる．ライプニッツ級数は小数点演算，フィボナッチ数列は整数演算が中心であることも関係していることが考えられる．ZYBO については，MIPS 性能から見れば leibniz では健闘しているものの，fib_multi でも良い結果を得ることができなかった．

次に，elixir_agg_csv の評価結果を**表 2** に示す．入力には [27] で公開されている 2 種類の CSV データを使用し，行見出しの数値はそれらの行数を示している．

本アプリケーションにおいても．ODROID-XU3 が最も良い結果となった．図 4 に示す通り Flow を積極的に活用しており，Elixir の並列性能を発揮できていることがわかる．このことから，高性能なコアを配置するよりもコア数を増やしたほうが性能面で有利という Elixir の特徴が IoT ボードでも活きてくることがわかる．なお今回は，表 1 では確認された RP3B 同士の OS 間の差異はみられなかった．

続けて，logistic_map について，Rustler の不使用時および使用時における評価結果を**表 3** および**表 4** に示す[*1]．行見出しの数値は Flow における並列数である．

これらの結果からは，NIF を介した他モジュールの呼び出しは性能向上に寄与できることが分かる．特に Flow における並列数の指定が小さい場合には，Rustler による性能向上が大きくなる．ただし，並列数が大きくなると Rustler

表 1 引数によるパラメータ固定時のライプニッツ級数計算およびフィボナッチ数列計算の実行時間［単位：秒］
Table 1 Evaluation results of Leibniz formula and Fibonacci number.

Board	leibniz	fib_simple	fib_multi
RP3B/Raspbian	114.470	1.076	14.474
RP3B/Ubuntu	118.107	1.223	15.888
ODROID-XU3	48.475	1.471	6.943
ZYBO	161.342	2.957	41.789

表 2 elixir_agg_csv の実行時間［単位：秒］
Table 2 Evaluation result of elixir_agg_csv.

Board	300_000	3_000_000
RP3B/Raspbian	11.128	81.776
RP3B/Ubuntu	9.326	82.487
ODROID-XU3	4.828	46.956
ZYBO	34.380	284.712

表 3 logistic_map の Rustler 不使用時の実行時間［単位：秒］
Table 3 Evaluation result of logistic_map without Rustler.

Board	1	2	4	8
RP3B/Raspbian	46.989	23.486	14.020	14.370
RP3B/Ubuntu	49.799	25.134	14.821	15.200
ODROID-XU3	23.884	12.726	8.895	6.865
ZYBO	87.184	47.905	47.942	48.387

表 4 logistic_map の Rustler 使用時の実行時間［単位：秒］
Table 4 Evaluation result of logistic_map with Rustler.

Board	1	2	4	8
RP3B/Raspbian	24.318	21.584	13.745	14.550
RP3B/Ubuntu	24.065	21.550	14.613	14.948
ODROID-XU3	13.769	9.922	7.619	6.920
ZYBO	50.701	40.436	38.454	39.290

表 5 phoenix-showdown の評価結果
Table 5 Evaluation result of phoenix-showdown.

Board	Throughput (req/s)	Latency (ms)	Consistency (σms)
RP3B/Raspbian	785.25	351.47	771.85
RP3B/Ubuntu	878.89	113.30	31.68
ODROID-XU3	859.60	112.67	25.82
ZYBO	459.28	216.72	83.10

[*1] [24] のうち benchmarks3 および benchmarks8 を使用した．

による性能向上が見込めなくなる．これは，NIF を介して実行される Rust モジュールは，Erlang VM による並列性能の恩恵を受けられないためであることが考えられる．

最後に，`phoenix-showdown` の評価結果を**表 5** に示す．ネットワーク越しのラップトップから `wrk` コマンドによって 4 スレッド，同時 100 コネクション，30 秒の負荷を掛けて評価を行った．評価指標は，Phoenix サーバに対するリクエストのスループット（秒ごとのリクエスト数），リクエストのレイテンシおよびその標準偏差（ミリ秒）である．

スループットについては RP3B/Ubuntu が最も良い結果となった．オンボードの WiFi モジュールによるネットワーク接続形態の利点を活かせているものと考えられる．OS の異なる RP3B+Raspbian ではやや劣る結果が得られた．また，USB アダプタモジュールによって無線化した ODROID-XU3 も遜色ない結果が得られている．ただし，ネットワークのトラフィックによる影響もあるため，これらはほぼ誤差の範囲内であるといえる．有線接続の ZYBO では，他の IoT ボードに勝る性能を発揮することはできなかった．Phoenix サーバの性能はネットワーク接続形態よりもコア性能のほうが重要になるものと考えられる．

5.3 追加評価の結果

本節では，ライプニッツ級数およびフィボナッチ数列のアプリケーションによっていくつかの追加評価を行う．

まず，**図 5** および**図 6** に，それぞれのアプリケーションに対する引数として `leibniz` の級数および `fib_simple` の項数を変更させたときの評価結果を示す．各図の軸は対数表示としている．`leibniz` では，級数に対して実行時間との綺麗な線形関係が確認できる．`fib_simple` では相関関係にややばらつきがあった．そして，項数が小さいときは ODROID-XU3 のほうが RP3B より優れるが，大きくなると優位性が逆転することとなった．`fib_simple` は再帰による処理であるため，カーネルによるメモリ管理がこの傾向に影響してくるものと考えられる．なお，ZYBO の `fib_simple` では，数列の数が 100_000 の時に heap メモリ不足による実行エラーが発生した．

次に，**図 7** に，`fib_multi` の入力データを 37 までの 10 個の数列として，並列プロセス数を変更させたときの評価結果を示す．この結果，RP3B および ZYBO については，それぞれの物理コア数である 4 および 2 まではプロセス数に比例して並列性能を発揮することが確認できた．8 コアを持つ ODROID-XU3 については，他の IoT ボードよりも優れた性能を示したものの，6 プロセスまでで並列性能の効果が頭打ちになっている．ODROID-XU3 に搭載されている Exynos 5 Octa 5422 では 4 コアの Cortex-15 と 4 コアの Cortex-A7 は別々のクラスタで構成され，L2 キャッシュを介して密結合されている．この SoC アーキテクチャの特性のためにコア数ほどの並列性能を発揮できなかったことが考えられる．

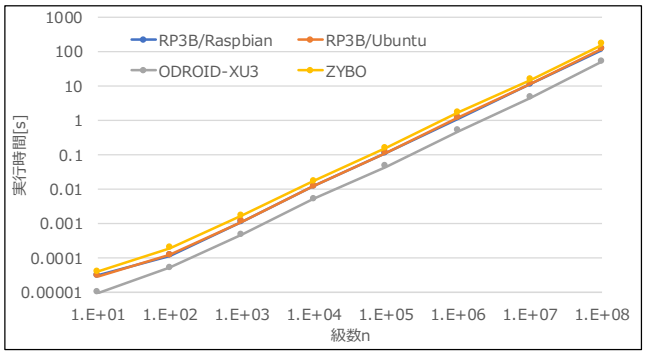

図 5 `leibniz` の級数変更時の実行時間
Fig. 5 Evaluation result of `leibniz`.

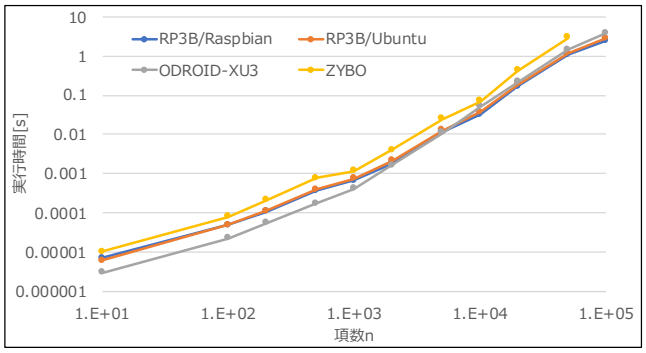

図 6 `fib_simple` の項数変更時の実行時間
Fig. 6 Evaluation result of `fib_simple`.

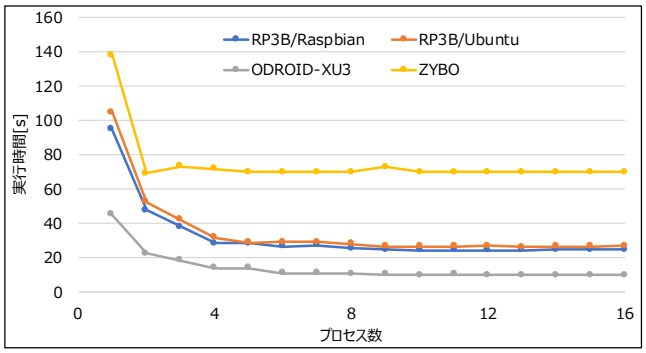

図 7 `fib_multi` の並列プロセス数変更時の実行時間
Fig. 7 Evaluation result of `fib_multi`.

最後に，RP3B および ODROID-XU3 に対して周波数制御を適用し，コアの動作周波数を調整して評価を行った．RP3B では 600MHz および 1.2GHz のいずれかに，

表 6 コアの動作周波数を 600MHz に固定した評価結果［単位：秒］
Table 6 Evaluation results when operation frequency is set to 600 MHz.

Board	leibniz	fib_simple	fib_multi
RP3B/Raspbian	226.617	2.478	28.906
RP3B/Ubuntu	240.612	2.723	31.856
ODROID-XU3	160.609	5.032	19.737
ZYBO	161.342	2.957	41.789

ODROID-XU3 では Cortex-A7 が 200MHz から 1.4GHz まで，Cortex-A15 が 2.0GHz までの 100MHz 刻みで，それぞれの動作周波数を設定できる．動作周波数の設定には Linux の `cpufrequtils` パッケージを用いた．**表 6** に，RP3B および ODROID-XU3 の動作周波数を 600MHz に固定した時の評価結果を示す．ZYBO の周波数は 650MHz である．各アプリケーションのパラメータは表 1 と同様である．`leibniz` の結果をみると，RP3B の性能が ZYBO よりも大きく低下することがわかった．また，ODROID-XU3 の性能も ZYBO とほぼ同等まで低下しており，`fib_simple` では ZYBO より劣る結果となった．このことから，コアの動作周波数のみならず命令セットの違いも Elixir の性能に大きな影響を及ぼすことがわかる．ただし，`fib_multi` で議論できる並列性能については，やはり ODROID-XU3 が優れる結果となった．

6. おわりに

本稿では，関数型言語 Elixir の IoT 分野における可能性に着目し，IoT ボード上における Elixir の基礎性能を評価した．Linux が動作する種々の IoT ボード上に Elixir の動作環境を整備し，様々な性能を評価するためのベンチマークアプリケーションを用意した．ベンチマーク評価によって得られた基礎性能より，IoT システム開発における Elixir の有用性を定量的に議論した．評価によって得られた知見を，下記のようにまとめる．

- Elixir の並列処理性能は IoT ボードにおいても発揮される．コアのマイクロアーキテクチャの優劣よりもコア数を重視すべきである．
- Elixir のプロセス単体の処理性能は，コアの命令セットアーキテクチャよりも動作周波数に強く依存する．
- Phoenix サーバについても IoT システムに十分な応答性能を示している．Phoenix の性能は，ネットワーク接続形態よりコア性能に依存する．

本稿では評価指標とはしなかったが，IoT システムにおいては消費電力も重要な要素である．これに関しては ZYBO が省電力性に優れている．実行性能については確かに劣るものの電力効率の面でも優位性を示す可能性がある．また，ZYBO については FPGA 部分の活用も行っていないが，これによって Elixir および Erlang VM の実行性能を高めることができれば，IoT ボード全体の性能向上に寄与できる可能性がある．

今後の大きな方向性としては，組込み環境に適した Elixir 実行環境の構築について検討している．今回のように Linux の動作する IoT ボードについてはエッジサーバに適用して Elixir の恩恵を享受することはできるが，より性能制約の厳しい IoT デバイスの開発にも Elixir を適用できる実行環境が望まれると考える．すなわち，micro-python や mruby のような組込み環境を Elixir でも用意できれば，開発生産性に貢献する波及効果は大きくなる．Raspberry Pi に限定すればすでに Nerves[28] や elixir_ale[29] などのデバイスプログラミング環境が登場しているが，組込みシステムにおいて重要視されるリアルタイム性については考慮されていない．

性能評価のベンチマークアプリケーションおよび評価用のツールキットは GitHub（`https://github.com/takasehideki/EEloT`）で公開している．他の IoT ボード上での定量的評価や組込み向けプログラミング言語との比較評価も必要であると考えており，最新の評価結果が得られたら GitHub においても公開していく予定である．

参考文献

[1] Lebedev, A.: What is an IoT platform? (online), `https://dzone.com/articles/what-is-an-iot-platform` (2018.06.07).
[2] Geogre, D.: MicroPython - Python for microcontrollers (online), `https://micropython.org/` (2018.06.07).
[3] Tanaka, K., Nagumanthri, A. and Matsumoto, Y.: mruby – Rapid Software Development for Embedded Systems, *Proc. of 15th Int'l Conf. on Computational Science and Its Applications (ICCSA)*, pp. 27–32, (2015).
[4] Thomas, D.: *Programming Elixir: Functional, Concurrent, Pragmatic, Fun* Pragmatic Bookshelf (2014).
[5] elixir-lang: Elixir is a dynamic, functional language designed for building scalable and maintainable applications https://elixir-lang.org. (online), `https://github.com/elixir-lang/elixir` (2018.06.07).
[6] Dean, J. and Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters, *Commun. ACM*, Vol. 51, No. 1, pp. 107–113 (2008).
[7] Flow: Computational parallel flows on top of GenStage (online), `https://github.com/elixir-lang/flow` (2018.06.07).
[8] Fedrecheski, G., Costa, L. C. P., and Zuffo., M. K.: Elixir programming language evaluation for IoT, *Proc. of IEEE Int'l Sympo. on Consumer Electronics (ISCE)*, pp. 105-106 (2016).
[9] Phoenix: Productive. Reliable. Fast. A productive web framework that does not compromise speed and maintainability (online), `http://phoenixframework.org` (2018.06.07).
[10] Mccord, C.: Elixir vs Ruby Showdown - Phoenix vs Rails (online), `https://littlelines.com/blog/2014/07/08/elixir-vs-ruby-showdown-phoenix-vs-rails` (2018.06.07).
[11] Raspberry Pi Foundation: Raspberry Pi 3 Model B - Raspberry Pi (online), `https://www.raspberrypi.org/products/raspberry-pi-3-model-b/` (2018.06.07).
[12] Raspberry Pi Foundation: Download Raspbian for Raspberry Pi (online), `https://www.raspberrypi.org/downloads/raspbian/` (2018.04.20).
[13] Ubuntu MATE: Ubuntu MATE for the Raspberry Pi 2 and Raspberry Pi 3 (online), `https://ubuntu-mate.org/raspberry-pi/` (2018.04.20).
[14] Samsung Inc.: Exynos 5 Octa 5422 Processor: Specs, Features (online), `http://www.samsung.com/semiconductor/minisite/exynos/products/`

mobileprocessor/exynos-5-octa-5422/ (2018.06.07).
[15] HardKernel Inc.: ODROID-XU3 (online), http://www.hardkernel.com/main/products/prdt_info.php?g_code=g140448267127 (2018.06.07).
[16] HardKernel Inc.: Index of /ubuntu_16.04lts (online), https://odroid.in/ubuntu_16.04lts/ (2018.04.20).
[17] アヴネット株式会社：ZYBO Zynq-7000 開発ボード (online), https://www.avnet.co.jp/kits/Digilent/ZYBO.aspx (2018.06.07).
[18] Xillybus Inc.: Xillinux: A Linux distribution for Zedboard, ZyBo, MicroZed and SocKit (online), http://xillybus.com/xillinux (2018.06.07).
[19] The Pragmatic Bookshelf: Source Code for Programming Elixir 1.2 (online), https://pragprog.com/titles/elixir12/source_code (2018.04.20).
[20] enpedasi: elixir_agg_csv (online), https://github.com/enpedasi/elixir_agg_csv (2018.04.20).
[21] BEAM VM Wisdoms: BIF and NIF functions ELI5 (online), http://beam-wisdoms.clau.se/en/latest/eli5-bif-nif.html (2018.06.07).
[22] hansihe: Safe Rust bridge for creating Erlang NIF functions (online), https://github.com/hansihe/rustler (2018.06.07).
[23] Miyazaki, T., et al.,: A Study of an Automorphism on the Logistic Maps over Prime Fields, *Proc. of The 2014 Int'l Sympo. on Information Theory and its Applications (ISITA2014)*, pp. 727–731 (2014).
[24] zeam-vm: LogisticMap (online), https://github.com/zeam-vm/logistic_map (2018.06.07).
[25] mroth: benchmark Sinatra-like web frameworks (online), https://github.com/mroth/phoenix-showdown (2018.04.20).
[26] Johansson, E., et al.,: HiPE: High-performance Erlang, *Technical Report ASTEC report 99/04*, Uppsala University (1999).
[27] enpedasi: CSV Files for Fukuokaex #5 session (online), https://github.com/enpedasi/fukuokaex5_csv (2018.04.20).
[28] Nerves Project: Craft and deploy bulletproof embedded software in Elixir (online), https://nerves-project.org (2018.06.07).
[29] fhunleth: Interact with hardware in Elixir - GPIOs, I2C and SPI (online), https://github.com/fhunleth/elixir_ale (2018.06.07).