JIT とコンパイラの動作を考慮した Kotlin 記述プログラムの性能向上に関する一考察

園山敦也[1] 柴田涼一[1] 佐藤悠祐[1] 神山剛[2] 福田晃[2] 小口正人[3] 山口実靖[1]

1. はじめに

Kotlin 言語が, 2011 年に Jetbrains によって開発され, 2016 年に公式にリリースされた[1]. Kotlin は Java との高い相互運用性を持つなどの特徴を有し，近年注目を集めている．

本稿では，Kotlin 記述プログラムの性能を Java 記述プログラムの性能と比較し，言語や言語処理系に依存した性能に関する考察を行う．

2. Kotlin

Kotlin は Jetbrains によって開発された静的型付けのプログラミング言語であり，オブジェクト指向と関数型プログラミングのスタイルをサポートしている．

Kotlin は JVM(Java Virtual Machine)環境をターゲットにして使用することができ，その場合は Kotlin で記述されたソースコードが Java バイトコードにコンパイルされ，そのバイトコードが JVM によって解釈，実行される．多くの場合 JVM 実装は JIT(Just-In-Time)コンパイラを搭載しており，バイドコード実行中には JIT コンパイラによるバイトコードのネイティブコードへの置き換え(コンパイル)が行われ，これにより高速化が達成されている．

3. 関連研究

Java 記述プログラムと Kotlin 記述プログラムの性能に関する考察としては，Schwermer による Kotlin プログラムと Java プログラムの性能評価[2]がある．メモリ消費量や GC(Garbage Collection)の動作などに関して詳細な評価が行われ，Java 言語プログラムの高速性などが示されている．ただし，JIT の動作や JIT の動作を考慮したバイトコード出力などに関する考察はなされていない．

4. 性能評価

本章にて, Kotlin と Java にてほぼ同一の(可能な限り近くした)単純な動作を行うプログラムを作成しその性能を比較し，Kotlin 記述プログラムと Java 記述プログラムの性能に関して考察を行う．具体的には，プログラムの基礎的な機能の一つである for 文(繰り返し文)の処理性能について考察を行う．

4.1 評価方法

図 1 の様な 1 億回空回りする for 文のマイクロベンチマークを Java と Kotlin で記述し，Windows の JVM 上で実行してその実行時間を計測した．使用した JVM は version 1.8.0_231, 使用コンパイラは Java が version 1.8.0_231, Kotlin が version 1.3.31 である．OS は Windows 10 Home，使用計算機の仕様は CPU Intel corei5-7200U，メモリ 8GB, SSD256GB である．計測は JIT が有効である状況と無効である状況の両方で行った．

```
Java:
for(int i=1;i<=100000000;i++){ }

Kotlin:
for(i:Int in 1..100000000){ }
```

図 1．Java と Kotlin の for 文空回しプログラム

4.2 実行時間

両言語で記述したプログラムの実行時間を図 2(JIT 有効)と図 3(JIT 無効)に示す．縦軸は for 文(1 億回)実行に要する時間の平均で，100 回の試行の平均である．

JIT 有効時の図 2 に着目すると，Java 記述プログラムの実行時間が 95%以上短く，極めて大きな実行時間の差が確認された．JIT 無効時の図 3 に着目すると，JIT 有効時の結果とは大きく異なり Java と Kotlin で大幅な実行時間の差は確認されず，Kotlin の方が 10%強高速である結果が得られれていることが分かる．図 3 にて大幅な速度差が確認されなかったことより，図 2 の大幅な速度差は JIT に起因するものであると考えられる．

4.3 ニーモニックの比較

両言語で記述された各プログラムをそれぞれのコンパイラでバイトコードにコンパイルして得られたバイトコードの内の for 文空回しに該当する部分を図 4 に示す．バイトコードはバイナリであり，人間には読むことができないため，図 4 に示すものは人間の読むことができるニーモニックに逆アセンブルされたものである．

図 4 からは，Java と Kotlin では中身が空の同じ for 文でもニーモニックが異なることが確認された。Kotlin では継

1 工学院大学
Kogakuin University
2 九州大学
Kyushu University
3 お茶の水女子大学
Ochanomizu University

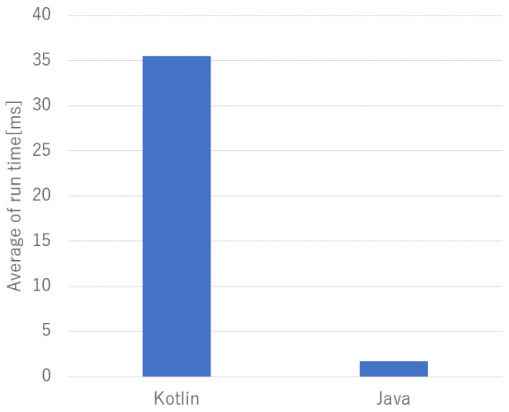

図 2. JIT 有効時の for 文空回し平均実行時間

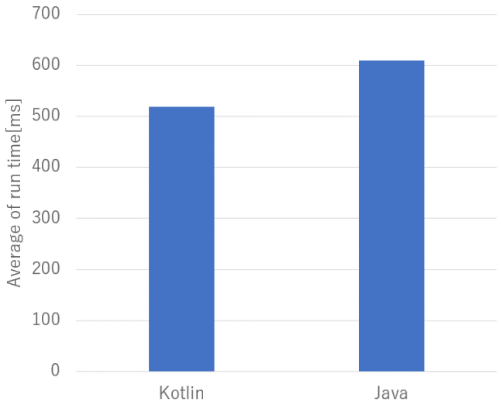

図 3. JIT 無効時の for 文空回し平均実行時間

```
Kotlin                          Java

L1                              L1
  LINENUMBER 3 L1                 LINENUMBER 5 L1
  ICONST_1                        ICONST_1
  ISTORE 2                        ISTORE 3
  LDC 100000000                  L2
  ISTORE 3                       FRAME APPEND [J I]
 L2                               ILOAD 3
 FRAME APPEND [J I I]             LDC 100000000
  ILOAD 2                         IF_ICMPGT L3
  ILOAD 3                         IINC 3 1
  IF_ICMPGT L3                    GOTO L2
 L4
  IINC 2 1
 L5
  GOTO L2
```

図 4. Java と Kotlin の for 文空回しのニーモニック

続条件の 1 億をあらかじめ局所変数に格納し呼び出すことで比較を行うが，Java では 1 億を毎回スタックへ積み比較を行っている. 先述の通り, 図 2 で確認された Java と Kotlin の大幅な性能差は JIT の最適化の具合によって生じていると考えられるが，この性能差はバイトコードに差があることで JIT の強い最適化が働く場合と働かない場合が生じているとが原因だと考えられる.

5. おわりに

本稿では，Kotlin と Java で記述されたマイクロベンチマークプログラムの性能を比較し，大きな性能差が生じることを示した．そして，両言語のプログラムよりそれぞれのコンパイラが生成するバイトコードを比較し，バイトコードに差があることを示した．今後は，バイトコード改変による JIT 発動の制御やそれによる性能向上手法について考察，関数型記述などの Kotlin 独自の記法のプログラムの性能評価などを行う予定である．

謝辞

本研究は JSPS 科研費 17K00109, 18K11277 の助成を受けたものである．

本研究は，JST，CREST JPMJCR1503 の支援を受けたものである．

参考文献

[1] Andrey Breslav. Kotlin 1.0 Released: Pragmatic Language for JVM and Android. https://blog.jetbrains.com/kotlin/2016/02/kotlin-1-0-released-pragmatic-language-for-jvm-and-android/ <accessed Dec. 3, 2019>.

[2] Schwermer, Patrik, “Performance Evaluation of Kotlin and Java on Android Runtime,” TRITA-EECS-EX; 2018:417, 2018.