A Study on Compiler Dependent Performance Improvement

Ryoichi Shibata
Kogakuin University
j016137@ns.kogakuin.ac.jp

Yusuke Sato
Kogakuin University
cm18019@ns.kogakuin.ac.jp

Masato Oguchi
Ochanomizu University
oguchi@is.ocha.ac.jp

Akira Fukuda
Kyushu University
fukuda@f.ait.kyushu-u.ac.jp

Takeshi Kamiyama
Kyushu University
kami@f.ait.kyushu-u.ac.jp

Saneyasu Yamaguchi
Kogakuin University
sane@cc.kogakuin.ac.jp

1 INTRODUCTION

An Android program written in Java or Kotlin language is compiled into Dalvik Executable (DEX) bytecode via Java bytecode. It then is executed on Android Runtime (ART). ART sometimes compiles DEX bytecode to native code using Just-In-Time (JIT) compiler. This paper evaluates the execution time of applications that are written in Java and Kotlin language without JIT compilation, analyzes the cause of the difference in performance based on DEX bytecode, and discusses a method for improving the performance by modifying DEX bytecode.

2 EVALUATION AND IMPROVEMENT

We implemented the almost same programs in Java and Kotlin, which execute an empty `for` statement times and are described in Fig. 1. Fig. 2 shows their times to complete the loop without JIT compilation. The results indicate that the bytecode from Kotlin is faster by 7.52%. Figure 3 and 4 show DEX bytecodes of the `for` statement generated from the Kotlin and Java source codes, respectively. The time of default bytecode from Java is slower than Kotlin, and this can happen if there is described an instruction that doesn't have to process every time in the target of `goto` instruction. We modified the bytecodes from the Java source code. Namely, we changed the target of `goto` instruction one instruction ahead. Figure 5 shows the modified bytecode. The result of "Java modified" in Fig. 2 shows its execution time. The results show that the performance of the bytecode from the Java language improved.

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

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

Figure 1: Source code of for statement in Java and Kotlin

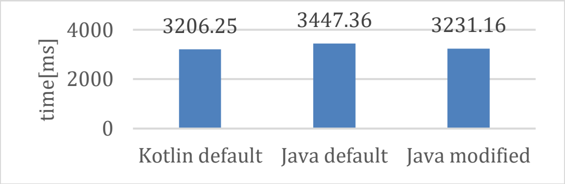

Figure 2: Execution time without JIT compilation

```
14d8c8: 1405 00e1 f505          |0012: const v5, #05f5e100
14d8ce: 1216                    |0015: const/4 v6, #1
14d8d0: 1217                    |0016: const/4 v7, #1
14d8d2: 3657 0400               |0017: if-gt v7, v5, +0004
14d8d6: b067                    |0019: add-int/2addr v7, v6
14d8d8: 28fd                    |001a: goto -0003
```

Figure 3: DEX bytecode of `for` statement from Kotlin

```
1240a0: 1215                    |0014: const/4 v5, #1
1240a2: 1406 00e1 f505          |0015: const v6, #05f5e100
1240a8: 3665 0500               |0018: if-gt v5, v6, +0005
1240ac: d805 0501               |001a: add-int/lit8 v5, v5, #01
1240b0: 28f9                    |001c: goto -0007
```

Figure 4: DEX bytecode of `for` statement from Java-(default)

```
11ae22: 1215                    |0013: const/4 v5, #1
11ae24: 1406 00e1 f505          |0014: const v6, #05f5e100
11ae2a: 3665 0500               |0017: if-gt v5, v6, +0005
11ae2e: d805 0501               |0019: add-int/lit8 v5, v5, #01
11ae32: 28fc                    |001b: goto -0004
```

Figure 5: Modified DEX bytecode that Java-derived

3 CONCLUTION

In this paper, we evaluated the performance of `for` statements written in Java and Kotlin language on ART. We showed their difference and a method for improving the performance by modifying the DEX bytecode generated from Java. We plan to implement this improving method in a Java compiler.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Numbers 17K00109, 18K11277.

This work was supported by JST CREST Grant Number JPMJCR1503, Japan.

REFERENCES

[1] Madhurima Banerjee, Subham Bose, Aditi Kundu, Madhuleena Mukherjee, "A comparative study: Java Vs kotlin Programming in Android," International Journal of Advanced Research in Computer Science, [S.l.], v. 9, n. 3, p. 41-45, june 2018. ISSN 0976-5697. doi: 10.26483/ijarcs.v9i3.5978.