等価プログラムのアセンブリ比較による Android DEX コンパイラの最適化性能テスト

Testing Optimized Performance of Android DEX Compiler by Comparing Assembly of Equivalent Programs

濱田 統弥 [1]　　石浦 菜岐佐 [2]
Toya Hamada　　Nagisa Ishiura

関西学院大学 理工学部 [1]
School of Science and Technology, Kwansei Gakuin Univ.

関西学院大学 工学部 [2]
School of Engineering, Kwansei Gakuin Univ.

1　はじめに

Android はモバイル用途を想定した OS であり, そのコンパイラの生成するコードの実行効率は重要な課題である. 吉田 [1] らは, 2 つのコンパイラが生成するコードを比較する差分法によって Android DEX コンパイラおよび AOT コンパイラの最適化不足を検出しているが, 両方のコンパイラに存在する最適化不足を検出できないという課題がある. 本稿では, 差分法で検出できない最適化不足の検出を図るため, テストプログラムとそれにソースコードレベルで最適化を施したプログラムから生成されるコードを比較する等価法に基づくテスト手法を提案する.

2　Android の最適化性能テスト

Android の処理系は, Java または Kotlin で書かれたプログラムを DEX コンパイラにより DEX コードに変換し, 仮想マシンでその一部を AOT コンパイラによりネイティブコードに変換する. 本手法では, DEX コンパイラと AOT コンパイラの組合わせで生成されるネイティブコードに期待する最適化が施されているかどうかをテストする.

文献 [1] の差分法では, ランダム生成した Java テストプログラムから生成される class ファイルを 2 つの異なる DEX コンパイラで DEX ファイルにコンパイルし, それぞれから AOT コンパイラにより生成される 2 つのネイティブコードを比較することにより最適化不足を検出している.

3　等価法を用いたランダムテスト

本稿で提案する等価法による最適化テスト手法の概要を図 1 に示す. ランダムに生成したテストプログラム (org.java) に対して, 期待される最適化をソースコードレベルで施した等価なプログラム (opt.java) を生成する. その 2 つを同じ DEX コンパイラと AOT コンパイラでコンパイルして得られる 2 つのネイティブコード (org.s と opt.s) の比較により, 最適化不足の検出を試みる.

本稿では, 定数畳み込みの最適化を対象とする. 図 2 に示すように, コンパイル時に値が決定できる部分式は定数に置き換える. 但し, volatile 修飾された変数やグローバル変数に対しては畳み込みを行わない.

4　実験結果

本手法に基づくテストシステムを Orange4 [2] を拡張する形で実装し, DEX コンパイラ d8 および x86_64 をターゲットとする AOT コンパイラの組合せに対してテストを行った. 結果を表 1 に示す. 最適化不具合が検出されたプログラムの一つを最小化したものと, それぞれ

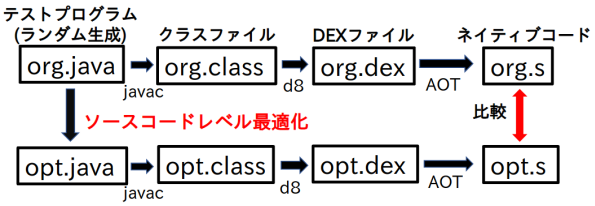

図 1: 等価法による最適化性能テスト

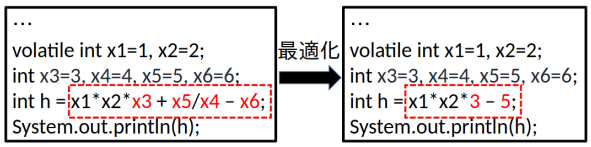

図 2: 畳み込み最適化の例

表 1: テスト結果

テスト数	最適化不足検出数	実行時間 (秒)
1000	156	3486

org.java

```
1:  class test{
2:    static volatile int x1 = 1;
3:    public static void main(String args[]){
4:      int x2 = 1;
5:      int x3 = 0;
6:      int h = (x3<<x2)*(1/x1);
7:      System.out.println(h);
8:    }
9:  }
```

org.s

```
      ...
1:  jz/eq +92 (0x000010bc)
2:  mov eax, 1
3:  cmp ecx, -1
4:  jz/eq +86 (0x000010c4)
5:  cdq
6:  idiv edx:eax, edx:eax / ecx
7:  mov eax, [RIP + 0xf89]
8:  cmpb [rax + 115], -16
9:  jb/nae/c +71 (0x000010c8)
10: test [rax + 7], 16
11: mov esi, [rax + 244]
      ...
```

opt.java

```
1:  class test{
2:    static volatile int x1 = 1;
3:    public static void main(String args[]){
4:      int x2 = 1;
5:      int x3 = 0;
6:      int h = 0*(1/x1);
7:      System.out.println(h);
8:    }
9:  }
```

opt.s

```
      ...
1:  jz/eq +75 (0x000010ab)
2:  mov eax, [RIP + 0xf9a]
3:  cmpb [rax + 115], -16
4:
5:
6:
7:
8:
9:  jb/nae/c + 67 (0x000010b3)
10: test [rax + 7], 16
11: mov esi, [rax + 244]
      ...
```

図 3: 最適化不足を検出したプログラムとアセンブリ

のアセンブリを図 3 に示す. org.s において, idiv 命令が生成されており, ソースコードの 6 行目を 0 に縮約する最適化が行われていないことがわかる.

5　むすび

本研究では等価法による Android DEX コンパイラの最適化性能テスト手法を提案した. 生成プログラムの強化が今後の課題である.

参考文献

[1] 吉田直生, 石浦菜岐佐: “ネイティブコード比較に基づく Android DEX コンパイラの最適化性能テスト,” 信学技報, VLD2021-74 (Jan. 2022).

[2] K. Nakamura and N. Ishiura: “Random testing of C compilers based on test program generation by equivalance transformation,” in *Proc. APCCAS 2016*, pp. 676–679 (Oct. 2016).