差分ランダムテストに基づくコンパイラの最適化機会の検出

Detecting Missed Oppotunities in Compiler Optimization by Differential Random Testing

岩辻光功 Mitsuyoshi Iwatsuji　橋本淳史 * Atsushi Hashimoto　石浦菜岐佐 Nagisa Ishiura

関西学院大学 理工学部 School of Science and Technology, Kwansei Gakuin University

1 はじめに

コンパイラには高い信頼性とともに高い最適化の性能が求められる．このため, コンパイラのテストにおいては, 生成されるコードが正しいかのみならず, 意図通りの最適化が行われているかどうかも重要なテスト項目になる．文献 [1] では, 2 つの等価なテストプログラムをコンパイルし, 得られたアセンブリコードを比較することによって最適化機会の逸失を検出している．しかしこの手法では, コンパイラが両方のプログラムに対して最適化を行えないことは検出できない．そこで本稿では, 差分テスト [2] に基づく最適化機会の検出手法を提案する．

2 C コンパイラランダムテストシステム Orange3

Orange3 [3] は, C コンパイラの算術最適化を対象としたランダムテストシステムであり, 図 1 のようなプログラムによって算術式 (13–14 行目) に対するコードが正しく生成されているかどうかをテストする．式数, 式の形, 演算子, 変数の型および初期値は設定に基づきランダムに決定される．ゼロ除算等の未定義動作は式生成時に回避される．

```
 1: #include <stdio.h>
 2: #define OK() printf("@OK@\n")
 3: #define NG(fmt,val) printf("@NG@ (test="fmt")\n",val)
 4: const signed int k8 = 144011145;
 5: int main (void)
 6: {
 7:   signed short x1 = 6;
 8:   static unsigned short x2 = 1U;
 9:   static signed long x4 = 4542636934L;
10:   volatile signed short x7 = -1;
11:   signed long t0 = -126191746930720794OL;
12:   signed long t1 = -42079927921L;
13:   t0 = (x2^(x4+((x7+k8)>>x2)));
14:   t1 = ((x4<<(x4/t0))/x1);
15:   if (t0 == 4614642507L) {OK();} else {NG("%d", t0);}
16:   if (t1 == 757106155L) {OK();} else {NG("%lld", t1);}
17:   return 0;
18: }
```

図 1　Orange3 が生成するプログラムの例

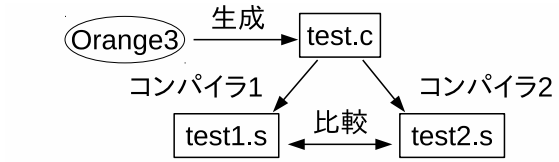

図 2　提案手法の流れ

3 差分テストに基づく最適化機会の検出

本稿では, 差分テスト [2] に基づいてコンパイラの最適化機会の逸失を検出する手法を提案する．図 2 に本手法の流れを示す．Orange3 で生成した C プログラム (test.c) を 2 つの異なるコンパイラ (または同じコンパイラの異なるバージョン) でコンパイルし, 生成される 2 つのアセンブリコード (test1.s, test2.s) を比較することにより最適化が行われているかをテストする．アセンブリコードの比較は命令数に基づいて行う．すなわち, test1.s と test2.s の命令数の小さい方を大きい方で割った値が閾値未満であれば, 最適化に差があると判定する．

差分を検出したプログラムは, 解析容易化のために最小化 (差分のある極力小さなものに縮約) する．最小化は, 式の削減や部分式の定数への置換等のプログラム縮約操作を差分が生じる限り繰り返すことにより行う．

表 1　実験結果

compiler (target) option	#test	#diff
GCC-5.0.0 (x86_64) -O3	62,672	593
Clang-3.7.0 (x86_64) -O3		

24 時間 (Ubuntu 14.10, Core i5-4200U 1.60GHz, RAM 7.7GB)

4 実験結果

本手法に基づくテストシステムを Perl で実装した．GCC-5.0.0 と Clang-3.7.0 (いずれも-O3 オプション) に対し, 閾値を 0.6 としてテストを 24 時間行った．結果を表 1 に示す．総テスト数は 62,672 であり, そのうち 593 個で差分を検出した．最小化したプログラムの一つを図 3 に示す．Clang が生成するコードの方が簡潔であり, GCC の最適化が不十分であることがわかる．本手法により検出した最適化機会の逸失は, GCC に 1 件 [1], Clang/LLVM に 2 件 [2] それぞれ報告している．

test.c

```
int main (void)
{
  volatile signed int x = 1;
  unsigned int t = ((unsigned int)1U<<x);
  if (t == 2U) ;
  else __builtin_abort();
  return 0;
}
```

test1.s (GCC-5.0.0 -O3)

```
main:
    subq  $24, %rsp
    movl  $1, %eax
    movl  $1, 12(%rsp)
    movl  12(%rsp), %ecx
    sall  %cl, %eax
    cmpl  $2, %eax
    jne   .L5
    xorl  %eax, %eax
    addq  $24, %rsp
    ret
.L5:
    call  abort
```

test2.s (Clang-3.7.0 -O3)

```
main:
    pushq %rax
    movl  $1, 4(%rsp)
    movl  4(%rsp), %eax
    cmpl  $1, %eax
    jne   .LBB0_2


    xorl  %eax, %eax
    popq  %rdx
    retq
LBB0_2:
    callq abort
```

図 3　最小化した差分プログラムの例

5 むすび

本稿では, 差分テストに基づくコンパイラの最適化機会検出方法を提案した．アセンブリコードの比較方法の改良が今後の課題である．

謝辞 本研究は一部 JSPS 科研費 25330073 の助成による．

参考文献

[1] 橋本, 石浦: “ランダムテストによる C コンパイラの算術最適化機会の検出,” 信学技報, VLD2014-139 (Jan. 2015).

[2] Robert B. Evans and Alberto Savoia: “Differential testing: A new approach to change detection,” in Proc. ACM ESEC-FSE ’07, pp. 549–552 (Sept. 2007).

[3] E. Nagai, A. Hashimoto, and N. Ishiura: “Reinforcing random testing of arithmetic optimization of C compilers by scaling up size and number of expressions,” IPSJ Trans. SLDM, vol. 7, pp. 91–100 (Aug. 2014).

*現在 株式会社野村総合研究所

[1] https://gcc.gnu.org/bugzilla/ id=66299

[2] https://llvm.org/bugs/ id=23673, 23672