# 目的コードの精査による COINS コンパイラ バックエンドの最適化の改良

米倉 翔一 †
Shoichi Yonekura
yonekur3@is.titech.ac.jp

佐々 政孝 †
Masataka Sassa
sassa@is.titech.ac.jp

† 東京工業大学大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

COINS コンパイラは、コンパイラ研究の基盤となる共通のコンパイラの作成を目的に開発された、並列化コンパイラ向け共通インフラストラクチャである。この COINS と GCC による同じ C プログラムに対する目的コードを精査し、COINS の出力するコードの時間効率が良くなるように COINS のバックエンドの最適化に改良を施した。具体的には、レジスタプロモーションにあったバグの修正、レジスタプロモーションで挿入されるストア命令の削減、整数定数による整数除算命令の乗算命令への変換の 3 つを行った。

## 1 はじめに

COINS [1] とは、コンパイラ研究の基盤となる共通のコンパイラの作成を目的に開発された、並列化コンパイラ向け共通インフラストラクチャである。COINS には SSA 最適化、レジスタプロモーションなどのさまざまな最適化が備えられており、出力する目的コードの時間効率も向上している。しかし GCC [2] の出力する目的コードと比較すると、実行時間に差があるものが多い。

COINS と GCC での SPEC CPU2000 [3] ベンチマークの測定結果を表 1 に示す。COINS では SSA 形式でのいくつかの最適化、命令スケジューリング、レジスタプロモーションを行っている。GCC では最適化オプション O2, O3 の 2 通りを行った。結果を見ると、一部を除き COINS と GCC で目的コードの時間効率に差があることが分かる。

この COINS と GCC の目的コードの時間効率の差を減らすため、それぞれの出力する目的コードを精査し、COINS の出力する目的コードの時間効率が良くなるように COINS のバックエンドの改良を行った。具体的には

- レジスタプロモーションにあったバグの修正
- レジスタプロモーションで挿入されるストア命令の削減
- 整数定数による整数除算の乗算命令への変換

の 3 つの改良を行った。このコードの精査には SPARC [4][5] の目的コードを用いた。

| | COINS | gcc-O2 | gcc-O3 |
|---|---|---|---|
| 164.gzip | 732 | 538 | 543 |
| 175.vpr | 639 | 492 | 487 |
| 181.mcf | 466 | 478 | 441 |
| 197.parser | 769 | 665 | 599 |
| 255.vortex | 704 | 558 | 551 |
| 256.bzip2 | 788 | 483 | 478 |
| 300.twolf | 1052 | 818 | 815 |
| 171.swim | 2093 | 1906 | 1904 |
| 172.mgrid | 1677 | 1943 | 1943 |
| 177.mesa | 752 | 655 | 644 |
| 179.art | 506 | 416 | 438 |
| 183.equake | 855 | 854 | 813 |

表 1: COINS と GCC での SPEC ベンチマークの実行結果 (単位は秒)

次章以降では、本研究で行った精査の流れ、改良の具体的内容、改良の効果について述べる。

## 2 目的コードの精査の流れ

本章では、本研究で行った目的コードの精査の流れの部分を説明する。

まず、精査を行う前に COINS と GCC を用いて、SPEC CPU2000 ベンチマークの測定を行った。測定

結果は前出の表 1 である。

SPEC ベンチマークの結果から、GCC とのベンチマークの実行時間の差が最も大きい 256.bzip2 について目的コードを精査することにした。

次に精査する範囲を絞るため、プログラムのプロファイルを用い、実行時間の長い関数について精査を行うことにする。しかし、現在 COINS にはプロファイルをとる機能がないため、GCC でのプロファイル結果を用いることにした。このプロファイルの結果より、実行時間の長い bzip2.c の generateMTFValues という関数の目的コードの精査を行った。

## 3 レジスタプロモーションのバグの修正

### 3.1 レジスタプロモーション

レジスタプロモーションとは、プログラムのコードの主にループの部分で普通はメモリに入っている値をレジスタに移して、コードの実行時間を改善することである [7]。COINS では、次の 2 つの条件

- ループ内に関数呼び出しがない。
- ループ内にポインタによるメモリへの間接参照がない

を満たすループについて、ループ内で参照されているグローバル変数に対してレジスタプロモーションを行う [6]。具体的には、そのループに入る前に、レジスタプロモーションを行うグローバル変数の値をレジスタにロードし、そのループを出たところでレジスタからそのグローバル変数に代入する。入れ子になっているループに対しては、上の条件を満足するループの中で一番外側のループを対象としている。

### 3.2 バグの出たコードとその修正

精査を行った関数 generateMTFValues のソースコードの一部を図 1 に示す。この図の (B) の while ループの部分の目的コードを図 2 に示す。

これを見ると、2, 3 行目の sethi, or の 2 命令で、変数 last のメモリアドレスを%i2 レジスタに入れ、6 行目の st 命令で%l1 レジスタの値を、メモリの%i2 レジスタに入れたアドレスの所にストアしている。同様にして、4〜5, 7〜12, 15 行目で, block, zptr, szptr という 3 つの変数のメモリアドレスにもレジスタの値をストアしている。しかしこの 4 つの変数はこの while ループ内で使用しておらず、これらの命令は不要である。しかもこのループは入れ子になっているループの最内ループになっており、実行時間に大きく影響があると思われる。

これらの命令はレジスタプロモーションのバグに

```
void generateMTFValues ( void )
     …
   for(i=0; i <= last; i++){     //(A)
         …
       while(ll_i != tmp){       //(B)
          j++;
          tmp2 = tmp;
          tmp = yy[j];
          yy[j] = tmp2;
       };
       …
       while(…){  …   }
       …
   }
}
```

図 1: 関数 generateMTFValues のソースコード

```
.L2113:
        sethi   %hi(last), %i2
        or      %i2,%lo(last),%i2
        sethi   %hi(block),%l7
        or      %l7,%lo(block),%l7
        st      %l1,[%i2]
        sethi   %hi(zptr),%i2
        or      %i2,%lo(zptr),%i2
        st      %l0,[%l7]
        sethi   %hi(szptr), %l7
        or      %l7,%lo(szptr),%l7
        st      %o0,[%i2]
        add     %i4,1,%i4
        mov     %i1,%i2
        st      %i0,[%l7]
        ldsb    [%i4],%i1
        stb     %i2,[%i4]
        and     %i1, 255, %i2
        cmp     %g2,%i2
        bne     .L2113
        add     %l6,1,%l6
```

図 2: 図 1 (B) の部分の目的コード

よって、レジスタプロモーションで図 1 の (A) の for ループの出口に挿入されるべきストア命令が、関連する一番外側のループの出口ではない図 1(B) の while ループの部分にも挿入されてしまったものである。このバグは、このような入れ子になっている外側のループの内側に、複数のループがある場合に発生する。改良の 1 つめとして、この不要な命令が挿入されるバグの修正を行った。

## 4 レジスタプロモーションで挿入するストア命令の削減

図 1 の (A) の for ループの出口の部分の目的コードを図 3 に示す。図 3 の 2〜13 行目はレジスタプロ

```
.L1595:
        sethi   %hi(last), %i2
        or      %i2, %lo(last), %i2
        sethi   %hi(block), %i1
        or      %i1, %lo(block), %i1
        st      %l1, [%i2]
        sethi   %hi(zptr), %i2
        or      %i2, %lo(zptr),%i2
        st      %l0, [%i1]
        sethi   %hi(szptr), %i1
        or      %i1, %lo(szptr),%i1
        st      %o0, [%i2]
        st      %i0, [%i1]
        cmp     %l5, 0
        ble     .L303
        sll     %o3, 1, %i4
```

図 3: 図 1 の (A) の for ループの出口部分の目的コード

モーションでループ出口に挿入するストア命令である。ここでは last, block, zptr, szptr の 4 つの変数についてのストア命令が挿入されている。しかし図 1 (A) の for ループ内では、この 4 つの変数の値は使用されてはいるものの、変更されてはいない。そのため、このループ出口に挿入するストア命令は不要である。そこで、レジスタプロモーションに対して、ループ内でプロモーションを行う変数の値が変更されているかどうかを調べ、値が変更されない変数に対しては、ループ出口にストア命令を挿入しないように改良を行った。

## 5 整数定数による整数除算命令の乗算命令への変換

図 4 に、ある整数を 2 で割ったときの COINS による目的コードを示す。COINS ではこのように sdiv と

```
sra     %i0,31,%g1      //%g1 ←%i0 を 31 ビット
                        //算術左シフト
mov     %g1, %y         //%y ←%g1
nop
nop
nop
sdiv    %i0, 2, %l1     //%l1 ←%i0/2
```

図 4: 整数を 2 で割ったときの目的コード

いう除算命令を用いて 2 での除算を行っている。しかし、2 のべき乗での整数定数除算（以後、定数除算と呼ぶ）は右シフト命令などを用いて計算できることがよく知られている。また、2 のべき乗以外での定数除算も乗算の上位 32 ビットをとる命令などで計算する方法が存在する。多くのコンピュータでは除算は非常に時間がかかるので、定数除算命令を避けるのは実行時間の向上に効果がある。そこで、Henry S. Warren, Jr. の著書 [8] にあるアルゴリズムを用いて定数除算命令を変換する最適化を実装した。

この変換により、図 4 の 2 での符号付除算は、被除数の符号を考え、図 5 のようになる。

```
srl     %i0,31,%l1      //%l1 ←%i0 を 31 ビット
                        //論理右シフト
add     %i0,%l1,%l1     //%l1 ←%i0+%l1
sra     %l1,1,%l1       //%l1 ←%l1 を 1 ビット
                        //算術右シフト
```

図 5: 最適化後の整数を 2 で割ったときの目的コード

2 のべき乗以外での定数除算の変換は 32 ビットマシンを対象とする。変換法は、$n/d$　を計算するとき、およそ $2^{32}/d$ であるような、除数 d の逆数の一種を n にかけて、積の左 32 ビットを取り出すことである。Henry S. Warren, Jr. のアルゴリズムを用いると、整数 n を 3 で割るコードは図 6 のようになる。ここで n は分子の整数が入っているレジスタ、t は一時レジスタ、q は商が入るレジスタである。この中の mulhs 命令は、32 ビット整数の乗算の結果 (64 ビットとなる) の上位 32 ビットをとる命令である。この命令は SPARC にはないが、乗算の結果の上位

```
mov     0x55555556, t   //t ← 0x55555556
mulhs   t, n, q         //q ← t*n の上位
                        //32 ビット
srl     n, 31, t        //t ← n を 31 ビット
                        //論理右シフト
add     q, t, q         //q ← t+q
```

図 6: 整数 n を 3 で割るコード

| | |
|---|---|
| base | SSA 最適化+命令スケジューリング +レジスタプロモーション (改良前) |
| no-bug | base にレジスタプロモーションの バグの除去を追加 |
| cut-store | no-bug にレジスタプロモーションの ストア命令の削減を追加 |
| change-div | cut-store に定数除算命令の 変換を追加 |
| gcc-O2 | GCC O2 オプション |
| gcc-O3 | GCC O3 オプション |

表 2: 実験時のコンパイルの種類

32 ビットは%y というレジスタに入るので、実装では乗算を行った後、この%y レジスタの中身を用いて実現した。

## 6 評価

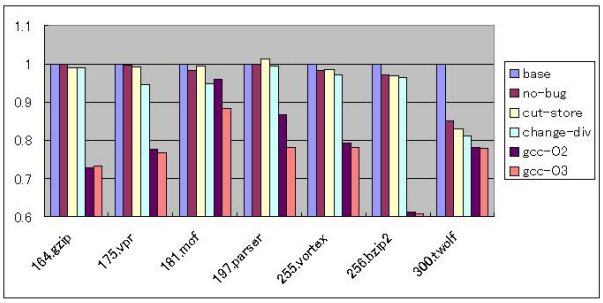
SPEC CPU2000 ベンチマークを用いて、本研究での改良を行った結果の目的コードの性能を測定した。実験には COINS version CVS (Date 2005/10/15 19:55), GCC version 4.0.2 を用い、Sun Microsystems の Sun Blade 1000 (UltraSPARC III, 750MHz × 2, メモリ 1GB) で行った。

実験では、表 2 の 6 種類でコンパイルを行った。

SPEC ベンチマークの結果を図 7 に示す。

図 7 を見るとバグの修正については 256.bzip2, 300.twolf などで実行時間が速くなっている。特に 300.twolf では 15%近く速くなっている。これは、これらのプログラムではバグの出るようなループ構造が多く、実行時間に影響が出ていたためと思われる。

ストア命令の削減については 300.twolf では効果が出ているが、そのほかでは余り効果が出なかった。これはストア命令の削減が適用できる箇所が少なかったことや、ループ出口のストア命令があまり実行時間に影響を与える部分ではなかったためだと考えら



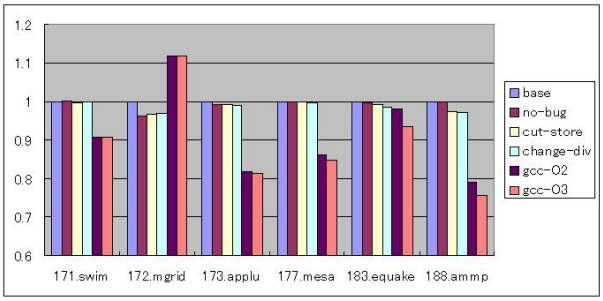




図 7: SPEC ベンチマークの実行結果 (base との相対値)

れる。

定数除算命令の変換は、175.vpr や 181.mcf ではある程度の効果があった。定数除算命令の多いプログラムではさらに効果が出ると思われる。

## 7 まとめ

COINS の出力する目的コードを精査し、その結果をもとに

- レジスタプロモーションにあったバグの修正
- レジスタプロモーションで挿入されるストア命令の削減
- 整数定数による整数除算の乗算命令への変換

の 3 つの改良を行った。改良の結果一部のプログラムでは 15%近く実行時間が改善された。

今後はさらに COINS の目的コードの実行時間が改善されるよう、今回扱った以外の部分にも改良を施したい。

### 参考文献

[1] COINS - Project. Coins - project home page. http://www.coins-project.org/.

[2] GCC. Gnu compiler collection home page. http://gcc.gnu.org/.

[3] SPEC. Standard performance evaluation corporation home page. http://www.spec.org/.

[4] SPARC International. Inc. 相越 克久, 田中長光 訳. SPARC アーキテクチャ・マニュアル バージョン 8. 株式会社トッパンジャパン, 1992.

[5] Sun Microsystems. Inc. *Ultra SPARC III Cu User's Manual*, 2004. http://www.sun.com/ processors/manuals/USIIIv2.pdf.

[6] 狩野祐介：素朴なレジスタプロモーションの実装・評価, 日本ソフトウェア科学会大会論文集, 第 22 回, 5B-3(2005).

[7] Lu, John and Keith D. Cooper. Register promotion in C programs. *PLDI '97: Proceedings of the ACM SIGPLAN 1997 Conference on Programing Language Design and Implementation.* ACM Press, 1997.

[8] ヘンリー・S・ウォーレン, ジュニア. ハッカーのたのしみ. 滝沢 徹, 鈴木 貢, 赤池 英夫, 葛 毅, 藤波 順久, 玉井 浩 訳, 株式会社エスアイビー・アクセス, 2004.

[9] 米倉翔一：目的コードの比較による COINS コンパイラ・バックエンドの改良, 東京工業大学理学部情報科学科卒業論文, 2006.