Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа программной инженерии

Лабораторная работа №4

Тема: Компилятор GCC. Оптимизация. Выбор оптимальной опции оптимизации

Работу выполнил студент
группы № в13534/22
_____ М. В. Андреев
«___» _____ 2019 г.

Работу проверил
_____ А. В. Петров
«___» _____ 2019 г.

Санкт-Петербург, 2019 г.

Постановка задачи

Цель работы – выбрать опции оптимизации для приложения. На основе примера, демонстрирующего различные уровни оптимизации, написать сценарий, выполняющий следующие действия в цикле:

- Компиляцию вашего приложения, не интерактивно обрабатывающего данные на языке C/C++/Fortran/Objective C/Objective C++/Ada с ключами оптимизации
  - -O0
  - -Os
  - -O1
  - -O2
  - -O3
  - -O2 -march=native
  - -O3 -march=native
  - -O2 -march=native -funroll-loops
  - -O3 -march=native -funroll-loops
- Вычисление времени выполнения программы (time). Приложение без оптимизации должно работать по меньшей мере 20 с.
- Вычисление занимаемого исполняемым файлом дискового пространства (в байтах) (du).

Ход выполнения работы

Для нахождения наиболее эффективной опции оптимизации была написана программа для перемножения двух матриц размером 1000x1000 элементов на языке C. Исходный код программы приведен в приложении. Время работы исполняемого файла без оптимизации составло 20,44 секунды, что удовлетворяет условию задачи.

Далее был составлен сценарий для компиляции исходной программы и вывода ее времени выполнения в секундах и размера в байтах Листинг сценария приведен в приложении. Результаты времени исполнения и размера исполняемого файла с использованием разных ключей оптимизации представленны в таблице 1.

Таблица 1. Время выполнения программы и ее размер в зависимости от ключа

Ключ оптимизации	Время исполнения, с	Размер, байт
-O0	20,44	8600
-Os	8,42	8600
-O1	9,55	8600
-O2	8,69	8600
-O3	8,39	8600
-O2 -march=native	9,31	8600
-O3 -march=native	9,75	8600
-O2 -march=native -funroll-loops	8,92	8600
-O3 -march=native -funroll-loops	8,64	8600

Было выявленно, что исполняемый фаил, скомпилированный с использованием ключа оптимизации -O3, исполняется быстрее всех, вследствие чего он был выбран в качестве оптимальной опции.

Далее была проведена оптимизация с оптимальной опцией и межпроцедурной оптимизацией и оптимизацией времени компоновки (-flto -fipa-*). Результаты времени исполнения и размера исполняемого файла представленны в таблице 2.

Таблица 2. Время выполнения программы и ее размер в зависимости от ключа с оптимизацией времени компоновки

Ключ оптимизации	Время исполнения, с	Размер, байт
-O3 -march=native -flto	8,49	8600
-O3 -march=native -fipa-reference	9,94	8600

Было выявленно, что исполняемый файл, скомпилированный с использованием ключа оптимизации -O3 -flto, выполняется дольше на 10 мс.

Далее была проведена оптимизация с оптимальной опцией и с оптимизацией с обратной связью (-fprofile-generate/-fprofile-use). Результаты времени исполнения и размера исполняемого файла представленны в таблице 3.

Таблица 3. Время выполнения программы и ее размер в зависимости от ключа с оптимизацией с обратной связью

Ключ оптимизации	Время исполнения, сек	Размер, байт
-O3 -fprofile-generate	7,83	79712
-O3 -fprofile-use	10,18	8600

Было выявленно, что исполняемый файл, скомпилированный с использованием ключа оптимизации -O3 -fprofile-generate, выполняется быстрее всех, однако и его размер существенно больше.

На рисунке 1 представлено соотношение времени работы исполняемого файла в зависимости от ключа оптимизации.

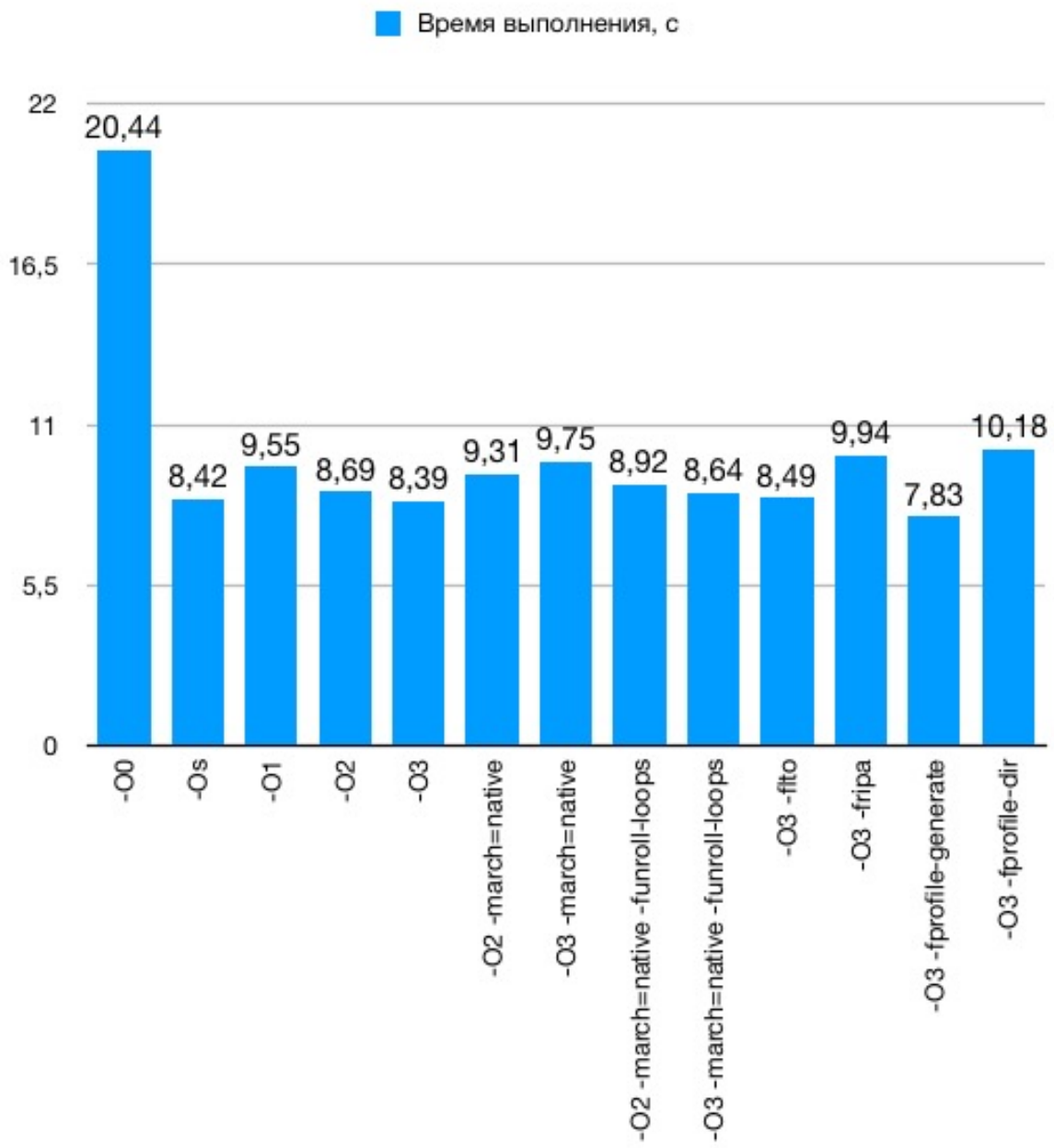

Рисунок 1. Время выполнения исполняемого файла в зависимости от ключа оптимизации

На рисунке 2 представлено соотношение размера исполняемого файла в зависимости от ключа оптимизации.

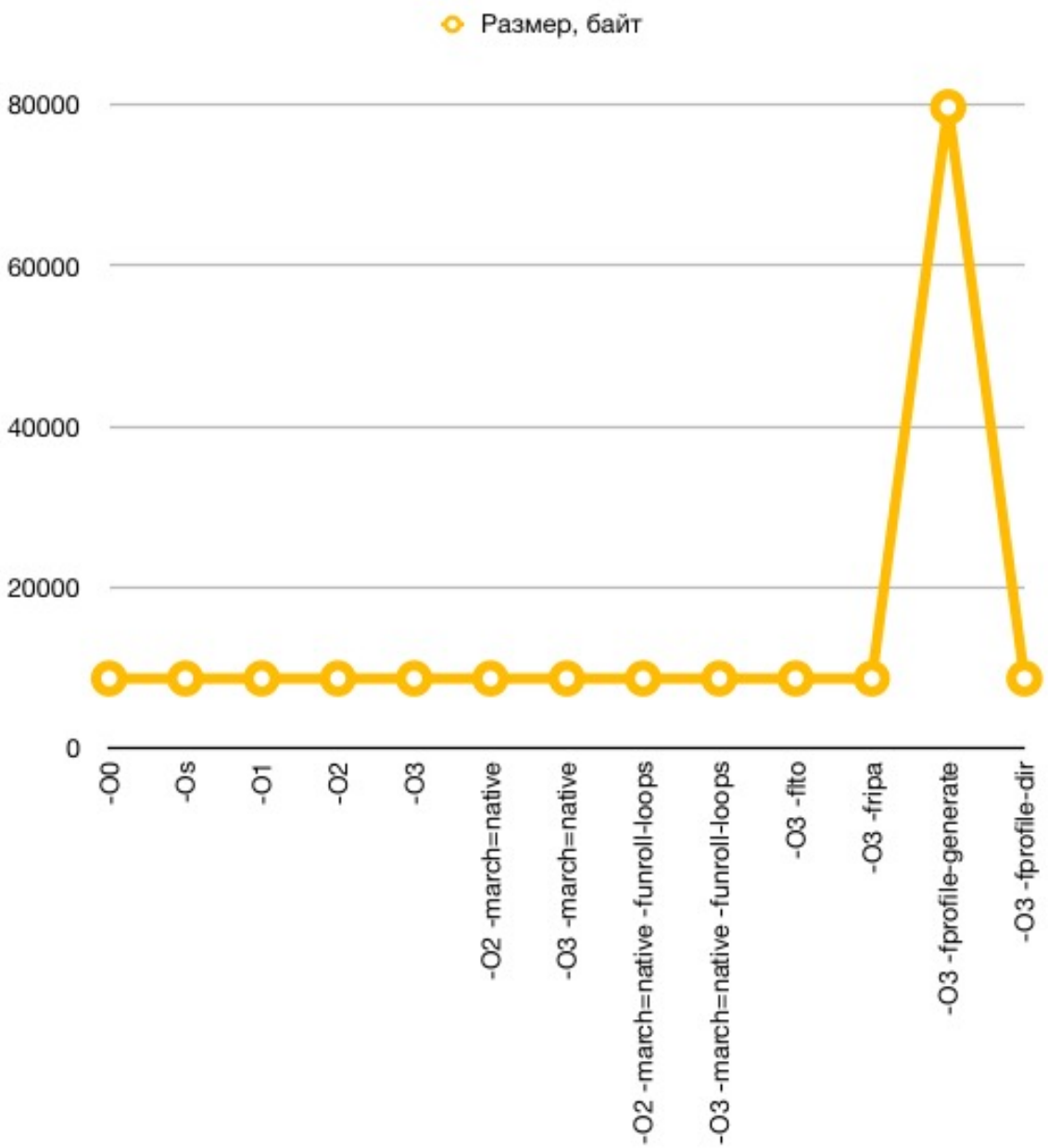

Рисунок 2. Размер исполняемого файла в зависимости от ключа оптимизации.

Заключение

Вывод: в результате выполнения лабораторной работы было выявленно, что размер файла не изменялся до использования оптимизации с обратной связью. Исполняемый файл, скомпилированный с ключом -O3 -fprofile-generate исполнился быстрее (7,83 с), чем исполняемые файлы, скомпилированные с другими ключами, однако его размер был существенно больше. Таким образом, в качестве опимальной была выбрана опция сключом -O3 (Время выполнения файла составило 8,39 с).

Приложение

Листинг 1. Произведение матриц

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 1000

int main() {
  int **A = (int**)malloc(N * sizeof(int*));
  int **B = (int**)malloc(N * sizeof(int*));
  int **C = (int**)malloc(N * sizeof(int*));

  int i, j, k;

  for(i = 0; i < N; i++) {
    A[i] = (int*)malloc(N * sizeof(int));
    B[i] = (int*)malloc(N * sizeof(int));
    C[i] = (int*)malloc(N * sizeof(int));
  }

  srand(time(NULL));
  for(i = 0; i < N; i++)
    for (j = 0; j < N; j++) {
      A[i][j] = rand() % 10;
      B[i][j] = rand() % 10;
    }

  for(i = 0; i < N; i++)
  for(j = 0; j < N; j++) {
    C[i][j] = 0;
    for(k = 0; k < N; k++)
      C[i][j] += A[i][k] * B[k][j];
    }

  // printf("\nmatrix A\n");
  // for(i = 0; i < N; i++) {
  //   for(j = 0; j < N; j++)
  //     printf("%d ", A[i][j]);
  //   printf("\n");
  // }

  // printf("\nmatrix B\n");
  // for(i = 0; i < N; i++) {
  //   for(j = 0; j < N; j++)
```

```
44    //      printf("%d ", B[i][j]);
45    //    printf("\n");
46    // }
47
48    // printf("\nthe result of multiplying\n");
49    // for(i = 0; i < N; i++) {
50    //    for(j = 0; j < N; j++)
51    //      printf("%3d ", C[i][j]);
52    //    printf("\n");
53    // }
54
55    for(i = 0; i < N; i++) {
56      free(A[i]);
57      free(B[i]);
58      free(C[i]);
59    }
60
61    free(A);
62    free(B);
63    free(C);
64
65    return 0;
66  }
```

Листинг 2. Сценарий

```
1  #!/bin/bash
2
3  name=$1
4
5  gcc -Wall -O0 $1
6  echo "-O0"
7  out="$(time ./a.out)"
8  by="$(du -b a.out)"
9  echo "byte :$by"
10
11 gcc -Wall -Os $1
12 echo "-Os"
13 out="$(time ./a.out)"
14 by="$(du -b a.out)"
15 echo "byte :$by"
16
17 gcc -Wall -O1 $1
18 echo "-O1"
19 out="$(time ./a.out)"
20 by="$(du -b a.out)"
21 echo "byte :$by"
```

```
22
23 gcc -Wall -O2 $1
24 echo "-O2"
25 out="$(time ./a.out)"
26 by="$(du -b a.out)"
27 echo "byte :$by"
28
29 gcc -Wall -O3 $1
30 echo "-O3"
31 out="$(time ./a.out)"
32 by="$(du -b a.out)"
33 echo "byte :$by"
34
35 gcc -Wall -O2 -march=native $1
36 echo "-O2 -march=native"
37 out="$(time ./a.out)"
38 by="$(du -b a.out)"
39 echo "byte :$by"
40
41 gcc -Wall -O3 -march=native $1
42 echo "-O3 -march=native"
43 out="$(time ./a.out)"
44 by="$(du -b a.out)"
45 echo "byte :$by"
46
47 gcc -Wall -O2 -march=native -funroll-loops $1
48 echo "-O2 -march=native -funroll-loops"
49 out="$(time ./a.out)"
50 by="$(du -b a.out)"
51 echo "byte :$by"
52
53 gcc -Wall -O3 -march=native -funroll-loops $1
54 echo "-O3 -march=native -funroll-loops"
55 out="$(time ./a.out)"
56 by="$(du -b a.out)"
57 echo "byte :$by"
58
59 gcc -Wall -O2 -march=native -flto $1
60 out="$(time ./a.out)"
61 by="$(du -b a.out)"
62 echo "-O2 -march=native -flto"
63 echo "byte :$by"
64
65 gcc -Wall -O2 -march=native -fipa-reference $1
66 out="$(time ./a.out)"
67 by="$(du -b a.out)"
```

```
68 echo "-O2 -march=native -fipa-reference"
69 echo "byte :$by"
70
71 gcc -Wall -O2 -march=native -fprofile-generate $1
72 out="$(time ./a.out)"
73 by="$(du -b a.out)"
74 echo "-O2 -march=native -fprofile-generate"
75 echo "byte :$by"
76
77 gcc -Wall -O2 -march=native -fprofile-use $1
78 out="$(time ./a.out)"
79 by="$(du -b a.out)"
80 echo "-O2 -march=native -fprofile-use"
81 echo "byte :$by"
82
83 gcc -Wall -O2 -march=native -flto -fprofile-use $1
84 out="$(time ./a.out)"
85 by="$(du -b a.out)"
86 echo "-O2 -march=native -flto -fprofile-use"
87 echo "byte :$by"
88
89 gcc -Wall -O2 -march=native -fipa-reference -fprofile-use $1
90 out="$(time ./a.out)"
91 by="$(du -b a.out)"
92 echo "-O2 -march=native -fipa-reference -fprofile-use"
93 echo "byte :$by"
94
95 gcc -Wall -O2 -march=native -flto -fprofile-generate $1
96 out="$(time ./a.out)"
97 by="$(du -b a.out)"
98 echo "-O2 -march=native -flto -fprofile-generate"
99 echo "byte :$by"
100
101 gcc -Wall -O2 -march=native -fipa-reference -fprofile-generate $1
102 out="$(time ./a.out)"
103 by="$(du -b a.out)"
104 echo "-O2 -march=native -fipa-reference -fprofile-generate"
105 echo "byte :$by"
```