Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Системное программное обеспечение GNU/Linux

Отчёт по лабораторной работе №4

Выполнил
студент гр. в3530904/00030 В.С. Баганов

Руководитель
ст. преподаватель, к.т.н. С.А. Федоров

«_____»_____ 202__ г.

Санкт-Петербург
2023

Содержание

Постановка задачи

Цель работы — перенесённое под целевую микроархитектуру ПО с оптимальными опциями оптимизации

1. Разработка требований к целевой и инструментальной платформе

- Аппаратная платформа: Виртуальная машина Parallels Desktop 16 Версия 16.1.2 (49151) с конфигурацией 4 Гб ОЗУ, с 4-мя процессорами, на MacBook, 2,6 GHz, 2-ядерный процессор Intel Core i5, 8 Гб ОЗУ. ☒
- Программная платформа: Linux debian-gnu-linux-10-vm 4.19.0-16-amd64, ядро SMP Debian 4.19.181-1 (2021-03-19) x86_64 GNU/Linux
- За основу для ТЗ взята программа, ввыполняющая умножение матрицы размером 4000*4000, заполненной случайными числами с плавающей точкой с диапазоне от 1 до 15 саму на себя и выполняющая поиск максимального значения в полученной новой матрице.
- Код реализован на языке программирования Fortran

2. Разработка ПО с однопоточным невекторизованным собственным (native) кодом согласно разработанному ТЗ

2.1. Код программы

```
program  laba_4 Baganov
   use Environment

   implicit none
   character(*), parameter :: input_file = "../data/input.txt",
    ↪  output_file = "output.txt"
   integer                 :: In = 0, Out = 0, N = 0, i = 0
   real(R_), allocatable   :: A(:, :), B(:, :)
   real(R_)                :: M


   open (file=input_file, newunit=in)
      read (in, *) N
      allocate (A(N, N))
      read (in, *) (A(i, :), i = 1, N)
   close (in)

   open (file=output_file, encoding=E_, newunit=Out)
      write (Out, '('//N//'f10.2')' (A(i, :), i = 1, N)
   close (Out)

   B = MATMUL(A, A)

```

```
    open (file=output_file, encoding=E_, newunit=Out, position='append')
       write (Out, '('//N//'f10.2)')' (B(i, :), i = 1, N)
    close (Out)

    M = MAXVAL(B)

   open (file=output_file, encoding=E_, newunit=Out, position='append')
       write (Out, "('MaxVal = ', f100.2)") M
    close (Out)

end program  laba_4 Baganov
```

2.2. Код программы исполняемого файла

```
#!/bin/bash
for opt in "-O3" "-Os" "-O1" "-O2" "-O3" "-O2 -march=native"
↪  "-O3 -march=native" "-O2 -march=native -funroll-loops"
↪  "-O3 -march=native -funroll-loops"
do
   echo "====================================="
   echo "Type: $opt"
   gfortran -Wall -std=f2008ts -static-libgfortran -flto -c
   ↪  src/environment.f90 -J obj/ -o obj/environment.o
   gfortran -Wall -std=f2008ts -static-libgfortran -flto $opt
   ↪  -ftree-vectorize -fopt-info-vec -c src/main.f90 -I obj/ -o
   ↪  obj/main.o
   gfortran -Wall -std=f2008ts -static-libgfortran -flto $opt
   ↪  -ftree-vectorize -fopt-info-vec -o bin/app ./obj/environment.o
   ↪  obj/main.o

   cd ./bin;
   aver_size=0
   for run in First Second Third Fourth Fifth
   do
   echo "====================================="
      time=$( TIMEFORMAT="%U"; { time ./app; } 2>&1 )
      size=$(du -sk ./app | cut -f1)
      echo "$aver_time"
      aver_size=$(echo $(($aver_size + $size)))
      echo "$run run; size: $size; time: $time"
   done

   echo "AVERAGE SIZE:"
   echo $(( $aver_size / 5 ))

   cd ..
   rm -rf obj/*
   rm -rf bin/*
done
```

2.3. Код программы исполняемого файла 2

```
#!/bin/bash
for opt in for opt in "-O2 -march=native -fipa-pta" "-O2 -march=native -flto" "-O2 -march=native -fipa-pta -flto" "-O2 -march=native -fprofile-generate" "-O2 -march=native -fprofile-use" "-O2 -march=native -fipa-pta -flto -fprofile-generate" "-O2 -march=native -fipa-pta -flto -fprofile-use" "-O2 -march=native -flto" "-O2 -march=native -flto" "-O2 -march=native -fipa-pta" "-O2 -march=native -fipa-pta -flto" "-O2 -march=native -fprofile-generate" "-O2 -march=native -fprofile-use" "-O2 -march=native -fipa-pta -flto -fprofile-generate" "-O2 -march=native -fipa-pta -flto -fprofile-use" "-O2 -march=native -ftree-vectorize"
do
   echo "===================================="
   echo "Type: $opt"
   gfortran -Wall -std=f2008ts -static-libgfortran -flto -c src/environment.f90 -J obj/ -o obj/environment.o
   gfortran -Wall -std=f2008ts -static-libgfortran -flto $opt -ftree-vectorize -fopt-info-vec -c src/main.f90 -I obj/ -o obj/main.o
   gfortran -Wall -std=f2008ts -static-libgfortran -flto $opt -ftree-vectorize -fopt-info-vec -o bin/app ./obj/environment.o obj/main.o

   cd ./bin;
   aver_size=0
   for run in First Second Third Fourth Fifth
   do
   echo "===================================="
      time=$( TIMEFORMAT="%U"; { time ./app; } 2>&1 )
      size=$(du -sk ./app | cut -f1)
      echo "$aver_time"
      aver_size=$(echo $(($aver_size + $size)))
      echo "$run run; size: $size; time: $time"
   done

   echo "AVERAGE SIZE:"
   echo $(( $aver_size / 5 ))

   cd ..
   rm -rf obj/*
   rm -rf bin/*
done
```

2.4. Тестирование и отладка однопоточного кода

Опорные данные: входящий файл input.txt содержащий размер квадратной матрицы и заполнение матрицы числами с плавающей точкой с диапазоне от 1 до 15. Размер файла 85.7M.

Время работы кода Тб/о, осуществляющего обработку данных без использования оптимизации: 56.32 с

Эффективность кода Рб/о, осуществляющего обработку данных без использования оптимизации:10000/24*56.32) = 7,39

Под эффективностью принять показатель P = Perf * 10000 / Comp, где Perf - обратное время работы кода на опорных данных (сек), Comp - сложность кода (число строк кода без комментариев).

3. Оценка влияния разных уровней оптимизации на эффективность разработанного приложения

В таблице ниже представлена компиляция разработанного приложения с ключами оптимизации: -O0, -Os, -O1, -O2, -O3, -O2 -march=native, -O3 -march=native, -O2 -march=native -funroll-loops, -O3 -march=native -funroll-loops. Для всех случаев оптимизация проводилась с отключенной векторизацией: -fno-tree-vectorize.

Таблица 3.1

Различные уровни оптимизации

Уровень опт-ии	Время,с	Эффективность	Размер, кбайт
-O0	36,17	11,51	356
-Os	36,64	11,37	356
-O1	36,62	11,37	356
-O2	36,13	11,52	356
-O3	36,02	11,56	356
-O2 march=native	34,81	11,96	356
-O3 -march=native	36,41	11,44	356
-O2 -march=native -funroll-loops	36,65	11,36	356
-O3 -march=native -funroll-loops	36,11	11,53	356

Вывод, дающий наибольшую производительность для разрабатываемого приложения является :-O2 march=native

4. Оценка влияния системных методов оптимизации на эффективность разработанного приложения

В дальнейшем будем использовать самый оптимальный уровень который получили :-O2 march=native и отключенной векторизацией.

4.1. Межпроцедурные методы оптимизации

Таблица 4.1

Межпроцедурные методы оптимизации

Метод опт-ии	Время,с	Эффективность	Размер, кбайт
-fipa-pta	59,11	7,04	356

4.2. Оптимизация времени компановки

Таблица 4.2

Оптимизация времени компановки

Метод опт-ии	Время,с	Эффективность	Размер, кбайт
-flto	57,46	7,25	356

4.3. Межпроцедурная и Оптимизация времени компановки

Таблица 4.3

Межпроцедурная и Оптимизация времени компановки

Метод опт-ии	Время,с	Эффективность	Размер, кбайт
-fipa-pta -flto	40,96	10,17	356

4.4. Оптимизация с обратной связью

Таблица 4.4

Оптимизация с обратной связью

Метод опт-ии	Время,с	Эффективность	Размер, кбайт
-fprofile-generate	40,85	10,19	356
-fprofile-use	50,64	8,22	356

4.5. Применение межпроцедурной оптимизации, оптимизациеи времени компоновки и оптимизации с обратной связью

Таблица 4.5

Комбинация методов оптимизации

Метод опт-ии	Время,с	Эффективность	Размер, кбайт
-fipa-pta -flto -fprofile-generate	43,63	9,54	356
-fipa-pta -flto -fprofile-use	36,96	11,27	356

Таблица 4.6

Системные методы оптимизации

Метод опт-ии	Время,с	Эффективность	Размер, кбайт
-flto	37,40	11,13	356
-fipa-pta	34,80	11,97	356
-fipa-pta -flto	34,51	12,07	356
-fprofile-generate	47,61	8,75	356
-fprofile-use	47,25	8,81	356
-fipa-pta -flto -fprofile-generate	44,88	9,28	356
-fipa-pta -flto -fprofile-use	45,69	9,11	356

Вывод, лучшие по времени оптимизации -flto -fipa-pta -fipa-pta -flto .

5. Оценка влияния векторизации на эффективность разработанного приложения

Оптимизация проводилась с оптимальным уровнем оптимизации полученным в предыдущей главе: -O2 march=native и с явной векториризациейй.

Таблица 5.1

Векторизация

Метод опт-ии	Время,с	Эффективность	Размер, кбайт
-ftree-vectorize	43,22	9,63	356

6. Оценка влияния выравнивания адресов на эффективность разработанного приложения

Для применения этого метода необходимо дополнительно написать в исходном коде следующую строку:

Таблица 6.1

!dir$ attribute align:n::array **Выравнивание адресов**

Это ведет к увеличению сложности кода (по количеству строк).

Метод опт-ии	Время,с	Эффективность	Размер, кбайт
-O3 -fno-tree-vectorize -fipa-pta -flto -fprofile-generate	75.95	5.28	356

Заключение

Целью работы было ознакомление с различными уровнями и методами оптимизации для разработки программы, эффективно задействующейсовременную целевую архитектуру.

Самая эффективная оптимизация -O2 march=native (-O2 большая часть доступных опций / march=native - оптимизация под используемую архитектуру). Лучшие по времени оптимизации.‘-flto’ (оптимизация по времени) ‘-fipa-pta’ и ‘-fipa-pta -flto’ . Векторизация не привела к снижению времени работы программымы.