Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации Сибирский Государственный Университет
Телекоммуникаций и Информатики СибГУТИ

Кафедра Вычислительных систем

Лабораторная работа №2
По дисциплине “Архитектура вычислительных систем”

Выполнил:
Студент группы ИВ-921
Гилев М.А.

Работу проверил:
Ассистент кафедры ВС
Петухова Я.В.

Новосибирск 2021

Задание

Реализовать программу для оценки производительности процессора (benchmark).

1. Написать программу(ы) (benchmark) на языке C/C++/C# для оценки производительности процессора. В качестве набора типовых задач использовать либо минимум 3 функции выполняющих математические вычисления, либо одну функцию по работе с матрицами и векторами данных с несколькими типами данных. Можно использовать готовые функции из математической библиотеки (math.h) [3], библиотеки BLAS [4] (англ. Basic Linear Algebra Subprograms — базовые подпрограммы линейной алгебры) и/или библиотеки LAPACK [5] (Linear Algebra PACKage). Обеспечить возможность в качестве аргумента при вызове программы указать общее число испытаний для каждой типовой задачи (минимум 10). Входные данные для типовой задачи сгенерировать случайным образом.
2. С помощью системного таймера (библиотека time.h, функции clock() илиgettimeofday()) или с помощью процессорного регистра счетчика TSC реализовать оценку в секундах среднего времени испытания каждой типовой задачи. Оценить точность и погрешность (абсолютную и относительную) измерения времени (рассчитать дисперсию и среднеквадратическое отклонение).
3. Результаты испытаний в самой программе (или с помощью скрипта) сохранить в файл в формате CSV со следующей структурой: [PModel;Task;OpType;Opt;InsCount;Timer;Time;LNum;AvTime;AbsErr;RelErr;TaskPerf], где

PModel – Processor Model, модель процессора, на котором проводятся испытания;
Task – название выбранной типовой задачи (например, sin, log, saxpy, dgemv, sgemm и др.);
OpType – Operand Type, тип операндов используемых при вычислениях типовой задачи;
Opt – Optimisations, используемы ключи оптимизации (None, O1, O2 и др.);
InsCount – Instruction Count, оценка числа инструкций при выполнении типовой задачи;
Timer – название функции обращения к таймеру (для измерения времени);
Time – время выполнения отдельного испытания; LNum – Launch Numer, номер испытания типовой задачи;
AvTime –Average Time, среднее время выполнения типовой задачи из всех испытаний[секунды];
AbsError – Absolute Error, абсолютная погрешность измерения времени в секундах;
RelError – Relative Error, относительная погрешность измерения времени в %;
TaskPerf – Task Performance, производительность (быстродействие) процессора при выполнении типовой задачи.

- * Оценить среднее время испытания каждой типовой задачи с разным типом входных данных (целочисленные, с одинарной и двойной точностью).
- ** Оценить среднее время испытания каждой типовой задачи с оптимизирующими преобразования исходного кода компилятором (ключи –O1, O2, O3 и др.).
- *** Оценить и постараться минимизировать накладные расходы(время на вызов функций, влияние загрузки системы и т.п.) при испытании, то есть добиться максимальной точности измерений.

4. Построить сводную диаграмму производительности в зависимости от задач и выбранных исходных параметров испытаний. Оценить среднее

быстродействие (производительность) для равновероятного использования типовых задач.

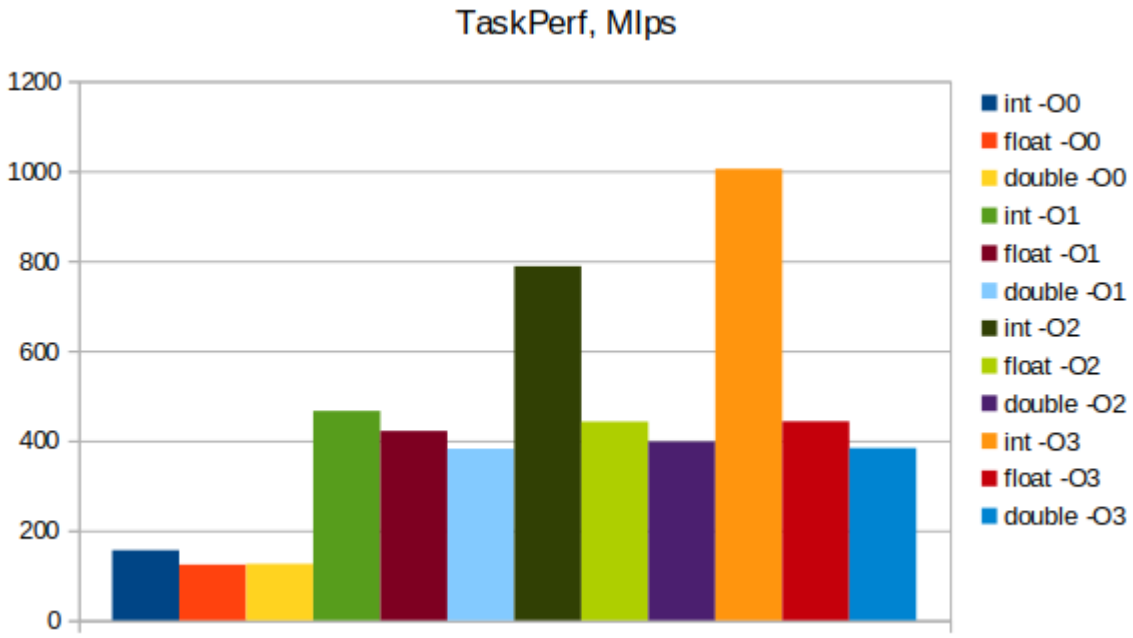
Результат выполнения

omega.txt

1. TimeBuf: 0.277 0.263 0.255 0.259 0.258 0.258 0.266 0.259 0.266 0.249
2. AvTime= 0.261sec.
3. W= 383.373
4. relErr=0.016sec.
5. absErr= 6.244%
6. Disp= 5.31175e-05
7. Deviation= 0.0073

file.csv

1. Model,Task,OpType,Opt,InsCount,Timer,Time,LNum,AvTime,AbsErr,RelError,TaskPerf
2. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;int;-O0;1e+08;wtime; 0.559;10; 0.641sec.;17.224%; 0.11sec.;156
3. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;float;-O0;1e+08;wtime; 0.802;10; 0.807sec.;7.416%; 0.06sec.;123.986
4. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;double;-O0;1e+08;wtime; 0.803;10; 0.796sec.;8.161%; 0.065sec.;125.585
5. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;int;-O1;1e+08;wtime; 0.223;10; 0.215sec.;10.97%; 0.024sec.;465.901
6. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;float;-O1;1e+08;wtime; 0.237;10; 0.237sec.;4.314%; 0.01sec.;421.16
7. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;double;-O1;1e+08;wtime; 0.257;10; 0.262sec.;11.393%; 0.03sec.;381.781
8. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;int;-O2;1e+08;wtime; 0.137;10; 0.127sec.;8.282%; 0.011sec.;787.988
9. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;float;-O2;1e+08;wtime; 0.245;10; 0.226sec.;8.118%; 0.018sec.;442.183
10. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;double;-O2;1e+08;wtime; 0.245;10; 0.252sec.;2.997%; 0.008sec.;397.484
11. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;int;-O3;1e+08;wtime; 0.104;10; 0.1sec.;10.248%; 0.01sec.;1004.86
12. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;float;-O3;1e+08;wtime; 0.223;10; 0.226sec.;1.899%; 0.004sec.;442.904
13. Intel(R) Pentium(R) CPU 4405U @ 2.10GHz;dgemm;double;-O3;1e+08;wtime; 0.258;10; 0.261sec.;6.244%; 0.016sec.;383.373


TaskPerf, MIps
1200
1000
800
600
400
200
0
int -O0
float -O0
double -O0
int -O1
float -O1
double -O1
int -O2
float -O2
double -O2
int -O3
float -O3
double -O3

Приложение

cpuTest.cpp

```
#include <cstdio>
#include <omp.h>
#include <sys/time.h>
#include <stdlib.h>
#include <inttypes.h>
#include <cmath>
#include <string>
#include <fstream>
#include <cstring>

#define COUNT_OPERATIONS 10000

double wtime()
{
	struct timeval t;
	gettimeofday(&t, NULL);
	return (double)t.tv_sec + (double)t.tv_usec * 1E-6;
}

template<typename T>
void matrix_vector_product(T *a, T *b, T *c, int m, int n) {
	for (int i = 0; i < m; i++) {
		c[i] = 0.0;
		for (int j = 0; j < n; j++)
			c[i] += a[i * n + j] * b[j];
	}
}

template<typename T>
double run_serial(int m, int n) {
	T *a, *b, *c;
	a = new T[n * m];
	b = new T[n];
	c = new T[m];
	for (int i = 0; i < m; i++) {
		for (int j = 0; j < n; j++)
			a[i * n + j] = i + j;
	}
	for (int j = 0; j < n; j++)
		b[j] = j;
	double t = wtime();
	matrix_vector_product(a, b, c, m, n);
	t = wtime() - t;
	delete(a);
	delete(b);
	delete(c);
	return t;
}

double maxElem(double* timeBuf) {
	double max = timeBuf[0];
	for (int i = 1; i < 10; i++) {
		if (max < timeBuf[i]) {
			max = timeBuf[i];
```

```
                }
        }
        return max;
}

double calcDisp(double* timeBuf, double averageTime) {
        double disp = 0;
        for (int i = 0; i < 10; i++) {
                disp += pow(timeBuf[i] - averageTime, 2);
        }
        return disp / 10;
}

double calcAverageTime(double* timeBuf) {
        double averageTime = 0;
        for (int i = 0; i < 10; i++)
                averageTime += timeBuf[i];
        averageTime /= 10;
        printf("Average time is %.3fsec.\n", averageTime);
        return averageTime;
}

double calcW(double* timeBuf, double averageTime) {
        double omega = (pow(COUNT_OPERATIONS, 2) / averageTime) / 1000000;
        printf("W = %.0f\n", omega);
        return omega;
}

void writeFile(double* timeBuf, double averageTime, double omega, double relErr, double absErr,
double disp, double otclon) {
        std::ofstream fd("omega.txt");
        fd << "TimeBuf:\t";
        for (int i = 0; i < 10; i++) {
                timeBuf[i] = round(timeBuf[i] * 1000) / 1000;
                fd << timeBuf[i] << '\t';
        }
        fd << "\nAvTime=\t" << round(averageTime * 1000) / 1000 << "sec.\nW=\t" << omega << '\
n';
        fd << "relErr=\t" << round(relErr * 1000) / 1000 << "sec.\nabsErr=\t" << round(absErr *
1000) / 1000
         << "%\nDisp=\t" << disp << "\nDeviation=\t" << round(otclon * 10000) / 10000 << '\n';
        fd.close();
}

int main(int argc, char **argv)
{
        int m, n;
        m = n = COUNT_OPERATIONS;
        double t;
        double timeBuf[10];
        //printf("Matrix-vector product (c[m] = a[m, n] mult b[n]; m = %d, n = %d)\n", m, n);
        if (argc == 2) {
                if (strcmp(argv[1], "int") == 0) {
                        for (int i = 0; i < 10; i++) {
                                t = run_serial<int>(m, n);
                                //printf("Test #%d: Elapsed time (serial): %.6f sec.\n", i
+ 1,t);
                                timeBuf[i] = t;
                        }
```

```
                }
                if (strcmp(argv[1], "double") == 0) {
                        for (int i = 0; i < 10; i++) {
                                t = run_serial<double>(m, n);
                                //printf("Test #%d: Elapsed time (serial): %.6f sec.\n", i
+ 1,t);
                                timeBuf[i] = t;
                        }
                }
                if (strcmp(argv[1], "float") == 0) {
                        for (int i = 0; i < 10; i++) {
                                t = run_serial<float>(m, n);
                                //printf("Test #%d: Elapsed time (serial): %.6f sec.\n", i
+ 1,t);
                                timeBuf[i] = t;
                        }
                }
                double averageTime = calcAverageTime(timeBuf);
                double omega = calcW(timeBuf, averageTime);
                double relErr = maxElem(timeBuf) - averageTime;
                double absErr = relErr / averageTime * 100;
                double disp = calcDisp(timeBuf, averageTime);
                double otclon = sqrt(disp);
                writeFile(timeBuf, averageTime, omega, relErr, absErr, disp, otclon);
        }
        return 0;
}
```

cpu.sh

```
#!/bin/bash


echo "Model,Task,OpType,Opt,InsCount,Timer,Time,LNum,AvTime,AbsErr,RelError,TaskPerf" > file.csv

model=$(lscpu | grep "Имя" | cut -b 24-)
task="dgemm"
OpType="int"
opt="none"
InsCount="1e+08"
Timer="wtime"
LNum=10

for ((idx=1;idx<13;idx++))
do
        if [[ $idx == 1 ]]
        then
                echo -e $(g++ -O0 -Wall -o test cpuTest.cpp -lm)
                opt="-O0"
        fi

        if [[ $idx == 4 ]]
        then
                echo -e $(g++ -O1 -Wall -o test cpuTest.cpp -lm)
                opt="-O1"
        fi

        if [[ $idx == 7 ]]
```

```
        then
                echo -e $(g++ -O2 -Wall -o test cpuTest.cpp -lm)
                opt="-O2"
        fi

        if [[ $idx == 10 ]]
        then
                echo -e $(g++ -O3 -Wall -o test cpuTest.cpp -lm)
                opt="-O3"
        fi

        if [[ $(($idx%3)) == 1 ]]
        then
                echo -e $(./test int)
                OpType="int"
                echo -e "./test int\n"
        fi

        if [[ $(($idx%3)) == 2 ]]
        then
                echo -e $(./test float)
                OpType="float"
                echo -e "./test float\n"
        fi

        if [[ $(($idx%3)) == 0 ]]
        then
                echo -e $(./test double)
                OpType="double"
                echo -e "./test double\n"
        fi

        out1=${model}";"${task}";"${OpType}";"${opt}";"${InsCount}";"${Timer}";"
        out2=$(cat omega.txt | grep "TimeBuf" | awk '{print $6}')";"${LNum}";"
        out3=$(cat omega.txt | grep "AvTime" | awk '{print $2}')";"$(cat omega.txt | grep
"absErr" | awk '{print $2}')";"
        out4=$(cat omega.txt | grep "relErr" | awk '{print $2}')";"$(cat omega.txt | grep "W" |
awk '{print $2}MIps')
        echo -e ${out1} ${out2} ${out3} ${out4} >> file.csv

done
```