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
Ярошев Р. А..

Работу проверил:
Ассистент кафедры ВС
Петухова Я.В.

Новосибирск 2021

Задание

Реализовать программу для оценки производительности процессора (benchmark).
1. Написать программу 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), библиотеки BLAS и/или библиотеки LAPACK. 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() или gettimeofday()) 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.

Результаты работы

```
TimeBuf:        0.105   0.104   0.104   0.105   0.104   0.105   0.105   0.104   0.107   0.104
AvTime   =      0.105sec.
W        =      954.26
relErr   =      2.004sec.
absErr   =      0.002%
Disp     =      6.44242e-07
Deviation=      0.0008
```

Рис.1. cpu.sh

Model	Task	OpType	Opt	InsCount	Timer	Time	LNum	AvTime	AbsErr	RelError	TaskPerf
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	int	-O0	100000000	wtime	0.233	10	0.232sec.	0.005%	1.954sec.	430.178
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	float	-O0	100000000	wtime	0.273	10	0.274sec.	0.004%	1.53sec.	365.31
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	double	-O0	100000000	wtime	0.254	10	0.253sec.	0.001%	0.47sec.	394.654
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	int	-O1	100000000	wtime	0.043	10	0.043sec.	0.001%	1.64sec.	2314.3
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	float	-O1	100000000	wtime	0.101	10	0.101sec.	0.001%	0.954sec.	986.702
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	double	-O1	100000000	wtime	0.107	10	0.107sec.	0.001%	0.871sec.	934.591
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	int	-O2	100000000	wtime	0.055	10	0.056sec.	0.003%	5.044sec.	1792.2
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	float	-O2	100000000	wtime	0.101	10	0.101sec.	0.001%	1.022sec.	987.282
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	double	-O2	100000000	wtime	0.106	10	0.107sec.	0%	0.318sec.	936.176
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	int	-O3	100000000	wtime	0.031	10	0.031sec.	0%	0.993sec.	3206.22
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	float	-O3	100000000	wtime	0.099	10	0.099sec.	0%	0.486sec.	1014.07
Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz	dgemm	double	-O3	100000000	wtime	0.104	10	0.105sec.	0.002%	2.004sec.	954.26

Табл.1. file.csv

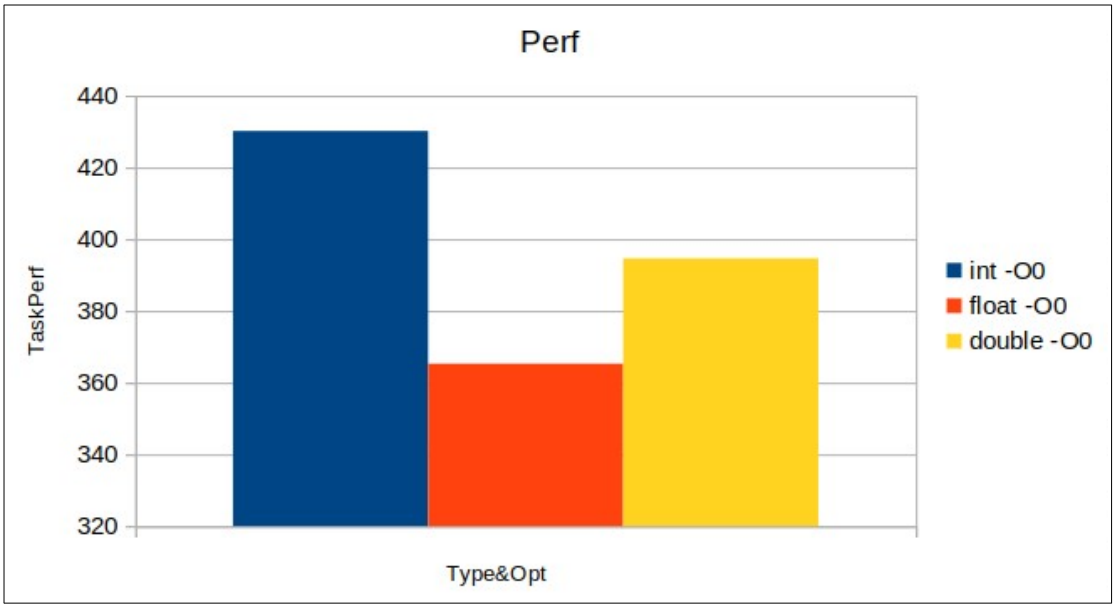

График 1. Perf

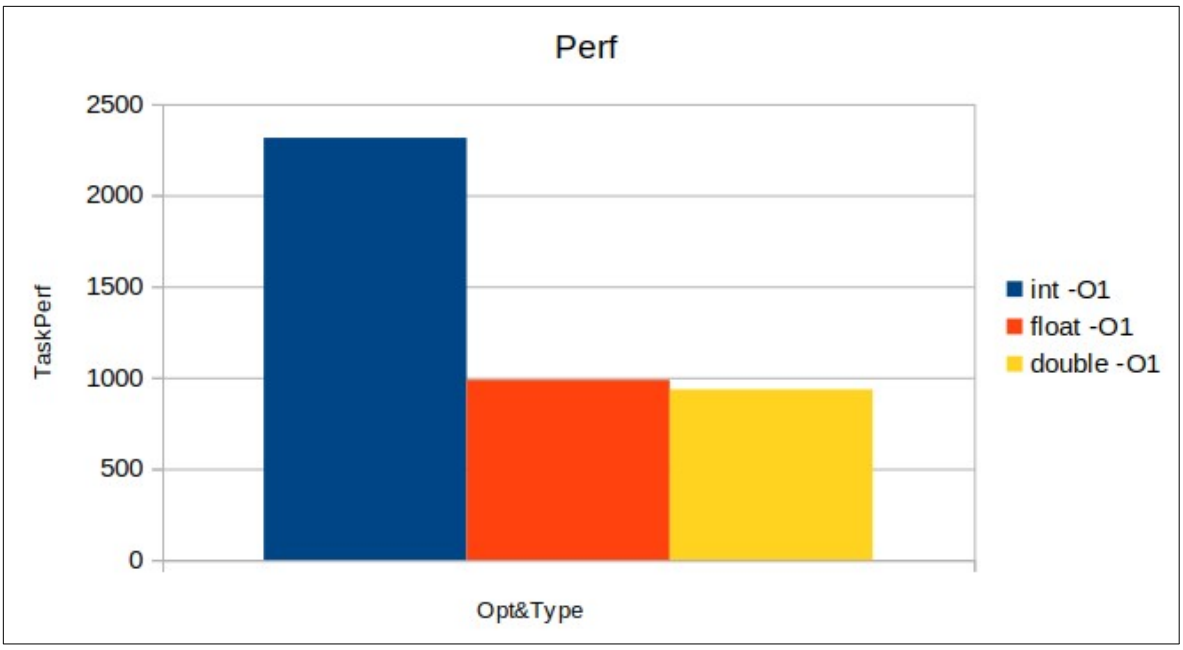

График 2. Perf

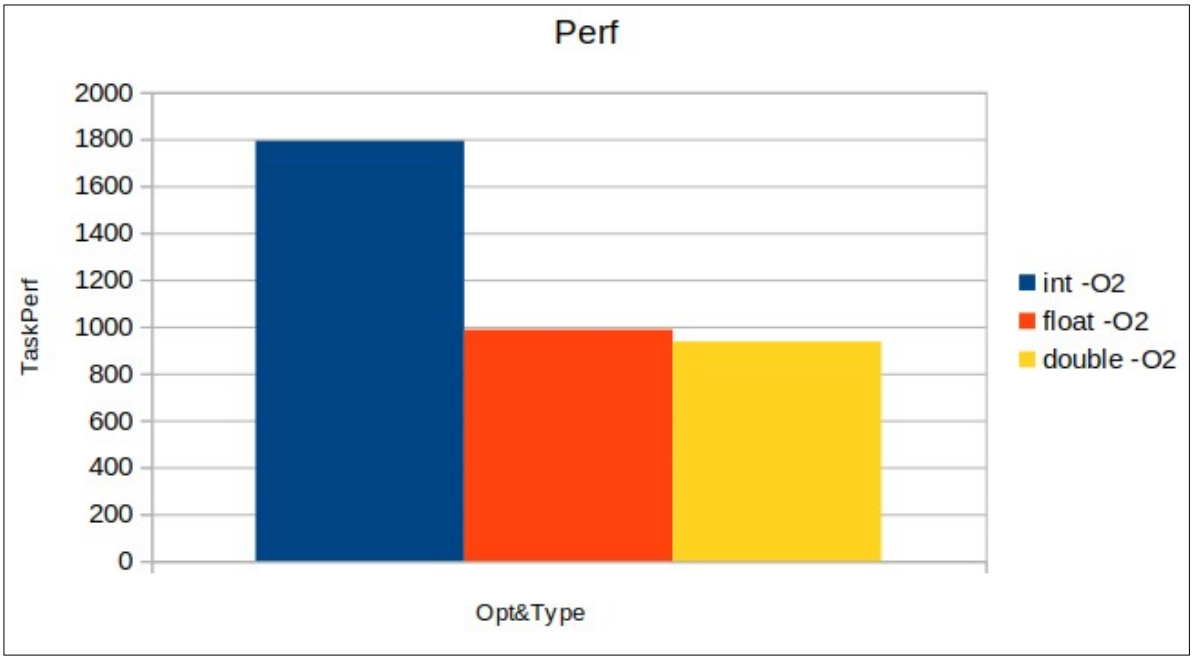

График 3. Perf

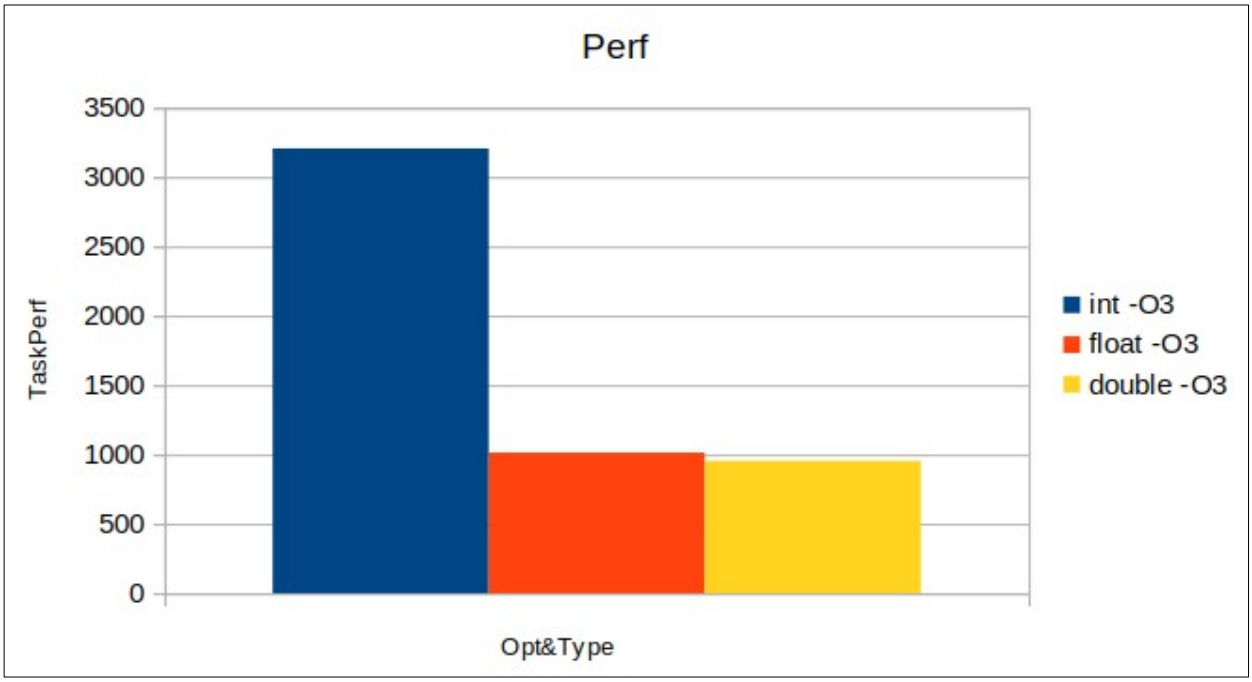

График 4. Perf

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 %.12fsec.\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];
        if (argc == 2) {
                if (strcmp(argv[1], "int") == 0) {
                        for (int i = 0; i < 10; i++) {
                                t = run_serial<int>(m, n);
                                timeBuf[i] = t;
                        }
                }
                if (strcmp(argv[1], "double") == 0) {
                        for (int i = 0; i < 10; i++) {
                                t = run_serial<double>(m, n);
                                timeBuf[i] = t;
                        }
                }
                if (strcmp(argv[1], "float") == 0) {
                        for (int i = 0; i < 10; i++) {
                                t = run_serial<float>(m, n);
                                timeBuf[i] = t;
                        }
                }
                double averageTime = calcAverageTime(timeBuf);
                double omega = calcW(timeBuf, averageTime);
                double absErr =fabs(maxElem(timeBuf) - averageTime);
                double relErr = absErr / 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,TaskPe
rf" > 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
```