Курс «Архитектура вычислительных систем». СибГУТИ. 2020 г.

Лабораторная работа No 2

Тема: Оценка производительности процессора.

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.
   Обеспечить возможность в качестве аргумента при вызовепрограммы 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.
4. Построить сводную диаграмму производительности в зависимости от задач и выбранных исходных параметров испытаний. Оценить среднее быстродействие (производительность) для равновероятного использования типовых задач.

Решение

1. Программа написана на языке C для оценки производительности процессора. В качестве типовой задачи была написана функция по умножению квадратных матриц.
   Входные данные из командой строки такие:
   Первое - название файла;
   Второе - размер квадратной матрицы;
   Третье - количество испытаний.

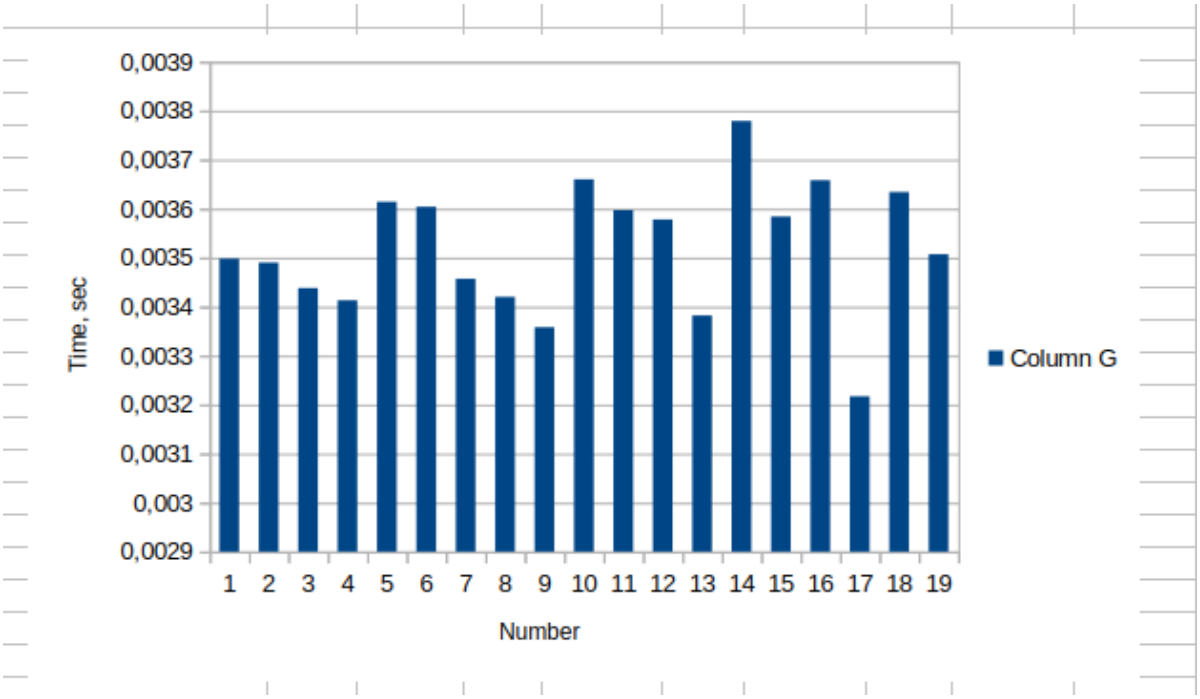
```
➜ src git:(develop) ✗ ./a.out 100 20
```

2. Измерение времени работы программы производилось с помощью библиотеки time.h и функции clock().

3. Результат программы сохранен в файл формата csv со следующей структурой:
   [PModel;Task;OpType;Opt;InsCount;Timer;Time;LNum;AvTime;AbsErr;RelErr;TaskPerf]

PModel	Task	OpType	Opt	InsCount	Timer	Time	LNum	AvTime	AbsErr	RelErr	TaskPerf
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003498	1	0.003498	0.000000	0.000000	285.877644
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,00349	2	0.003494	0.000004	0.114613	286.204923
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003438	3	0.003475	0.000037	1.085903	287.742183
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003413	4	0.003460	0.000047	1.369763	289.038225
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003614	5	0.003491	0.000123	3.414499	286.483699
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003604	6	0.003510	0.000094	2.622087	284.940875
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003457	7	0.003502	0.000045	1.301707	285.551114
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,00342	8	0.003492	0.000072	2.097953	286.389346
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003358	9	0.003477	0.000119	3.540467	287.613448
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,00366	10	0.003495	0.000165	4.502732	286.106661
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003597	11	0.003504	0.000093	2.572851	285.351112
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003578	12	0.003511	0.000067	1.884200	284.852945
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003382	13	0.003501	0.000119	3.509530	285.657782
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003779	14	0.003521	0.000258	6.838544	284.044798
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003584	15	0.003525	0.000059	1.651786	283.704040
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003658	16	0.003533	0.000125	3.413751	283.035556
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003217	17	0.003515	0.000298	9.248661	284.533115
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003634	18	0.003521	0.000113	3.104935	283.996781
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003507	19	0.003520	0.000013	0.382693	284.056931
11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz	matrix	int	None	1	clock	0,003635	20	0.003526	0.000109	2.994498	283.595423

4. Построена диаграмма на основе данных из пункта 3



Код

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

char Pmodel[1000];
char Pmodel_new[1000] = "\0";
int LNum = 1;
int fcF = 0;
double avgTimee = 0.0;
double avgTimee_o = 0.0;
double absErr = 0.0;
double relErr = 0.0;
double taskPef = 0.0;

void intMatrix(int rep, int N);
void PModel();
void csvFile(double time_spent);
void outEnd();
void avgTime();
void absError(double time_spent);
void RelError(double time_spent);
void TaskPerf();

int main(int argc, char **argv) {
  if (argc < 3) {
    printf("Check!\n");
    return 0;
  }
  PModel();
  csvFile(1);
  char *c;
  int rep = strtol(argv[2], &c, 10);
  int N = strtol(argv[1], &c, 10);

  intMatrix(rep, N);
}

void intMatrix(int rep, int N) {
  for (int i = 0; i < rep; i++) {
    double time_spent = 0.0;
    fcF = 1;
    clock_t begin = clock();
    int A = (int )malloc(N * sizeof(int *));
    int B = (int )malloc(N * sizeof(int *));
    int C = (int )malloc(N * sizeof(int *));
    for (int 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 (int i = 0; i < N; i++)
      for (int j = 0; j < N; j++) {
        A[i][j] = rand() % 100;
        B[i][j] = rand() % 100;
      }
    for (int i = 0; i < N; i++)
      for (int j = 0; j < N; j++) {
        C[i][j] = 0;
        for (int k = 0; k < N; k++) C[i][j] += A[i][k] * B[k][j];
      }
    for (int i = 0; i < N; i++) {
      free(A[i]);
      free(B[i]);
      free(C[i]);
    }
    free(A);
    free(B);
    free(C);
    clock_t end = clock();
    time_spent += (double)(end - begin) / CLOCKS_PER_SEC;
    printf("%f\n", time_spent);
    avgTimee += time_spent;
    // if (i == rep - 1) {
    //   fcF = 2;
    // }
    csvFile(time_spent);
  }
}

void csvFile(double time_spent) {
  FILE *file;
  file = fopen("csvFile.csv", "a");
  if (file == NULL) {
    printf("Error!\n");
    exit(0);
  }
  if (!fcF) {
    fprintf(file,
            "PModel;Task;OpType;Opt;InsCount;Timer;Time;LNum;AvTime;AbsErr;"
            "RelErr;TaskPerf\n");
  }
  // if (fcF == 2) {
  //   absError(time_spent);
  //   avgTime();
  //   fprintf(file, "%s;matrix;int;None;?InsCount?;clock;%f;%i;%f;%f\n",
  //   Pmodel,
  //           time_spent, LNum, avgTimee, absErr);
```

```
  // }
  if (fcF) {
    outEnd();
    avgTime();
    absError(time_spent);
    RelError(time_spent);
    TaskPerf();
    fprintf(file, "%s;matrix;int;None;1;clock;%f;%i;%f;%f;%f;%f\n", Pmodel,
            time_spent, LNum, avgTimee_o, absErr, relErr, taskPef);
    LNum++;
  }
  fclose(file);
}

void avgTime() { avgTimee_o = avgTimee / LNum; }

void absError(double time_spent) {
  absErr = fabs((avgTimee / LNum) - time_spent);
}

void RelError(double time_spent) { relErr = absErr * 100 / time_spent; }

void TaskPerf() { taskPef = 1 / avgTimee_o; }

void outEnd() {
  for (long unsigned int i = 0; i < strlen(Pmodel); i++) {
    if (Pmodel[i] == '\n') {
      Pmodel[i] = '\0';
      break;
    }
  }
}

void PModel() {
  FILE uname;
  int lastchar;
  uname = popen(
      "lscpu | grep -E '^Имя модели|^Model name' | sed 's/Имя "
      "модели:\s//;s/Model name:\s*//'",
      "r");
  lastchar = fread(Pmodel, 1, 1000, uname);
  Pmodel[lastchar] = '\0';
  printf("%s", Pmodel);
  pclose(uname);
}
```