COMPUTAÇÃO DE ALTO DESEMPENHO - PRIMEIRA LISTA DE EXERCÍCIOS

SÉRGIO CORDEIRO

1. Escreva um programa que gere uma sequencia 2500 números aleatórios inteiros com distribuição uniforme no intervalo de 0 a 255 divididos em oito faixas e calcule a frequência absoluta de incidência destes números em cada faixa.
Ex.:
Faixa 0 a 31 - Frequência = 309
Faixa 32 a 63 - Frequência = 305
Faixa 64 a 95 - Frequência = 315
Faixa 96 a 127 - Frequência = 327
Faixa 128 a 159 - Frequência = 307
Faixa 160 a 191 - Frequência = 299
Faixa 192 a 223 - Frequência = 317
Faixa 224 a 255 - Frequência = 321
Gere agora uma seguência de 25000 números aleatórios e divididos também em 8 faixas. Aumente este problema nesta proporção e verifique se há redução do tempo de execução utilizando os diversos níveis de otimização mostrados em sala de aula. Utilize o comando “time” do Linux para lhe auxiliar sua análise. Comente seus resultados e entregue uma listagem do seu código.

Código:

Listing 1. gcrand.c

```c
/*
gcrand.c
Gera sequência de números aleatórios e calcula a frequência absoluta de incidência em cada faixa de valores.
Uso:
  gcrand size
  onde size é o tamanho da sequência
Testado em GNU C sobre Linux (Ubuntu 12.04.5).
*/

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

#define LIMITE_SEQ      255
#define LIMITE_FAIXA    32
#define NUM_FAIXAS      8
```

```c

int main(int argc, char * argv[]) {
  // Obtém o tamanho da sequência
  int size = atoi(argv[1]);
  if ( size <= 0 ) {
    printf("Tamanho da sequencia deve ser positivo!\n");
    return 1;
    }
  // Inicializa o gerador de números aleatórios
  time_t t;
  srand((unsigned) time(&t));
  // Inicializa os contadores
  int contador [] = {0, 0, 0, 0, 0, 0, 0, 0};
  // Gera os valores e conta a frequência
  // não é preciso armazenar o valor
  for (int idx = 0; idx < size; ++idx) {
    int val =  rand() % LIMITE_SEQ;
    int pos = val / LIMITE_FAIXA;
    contador[pos]++;
    }
  // Imprime o resultado
  for (int idx = 0; idx < NUM_FAIXAS; ++idx) {
    printf("Faixa %d: %d valores\n",idx+1,contador[idx]);
    }
  return 0;
  }
```

Resultados (tempos em μs):

O	Tempo de parede		Tempo de usuário	
	n = 2500	n = 25000	n = 2500	n = 25000
0	469	47	485	485
1	469	47	438	438
2	313	31	360	360
3	47	31	375	375

Interpretação:
O programa não faz chamadas ao sistema operacional, por isso o tempo de núcleo é sempre nulo. A utilização de níveis de otimização crescentes gerou sempre resultados mensuráveis, às vezes no tempo de parede, às vezes no tempo de usuário.

2. Converta os valores a seguir para a representação de ponto flutuante apresentada em aula. a) +0,00565 b) - 674,25

a)

0,00565		
0	01110111	01110010010001110100011
0	77x	3923A3x

b)

- 674,25		
1	10001000	01010001001000000000000
1	88x	289000x

---

3. Realize a operação de multiplicação Matriz X Matriz para que se atinja 1 Mflop e 1 Gflop. Dimensione suas matrizes e realize quantas operações forem necessárias até atingir esta marca.

Para 1 Mflop, a melhor aproximação encontrada foi multiplicar uma matriz de dimensão 86 x 77 por outra de dimensão 77 x 76, dando como resultado uma matriz 86 x 76.
Para 1 Gflop, a melhor aproximação á multiplicar uma matriz de dimensão 872 x 752 por outra de dimensão 752 x 763, dando como resultado uma matriz 872 x 763.

---

4. Suponha que um processamento utilize muitas tarefas com ponto flutuante, sendo que 44% do tempo de execução consumido com esta tarefa. Qual o fator de agilidade requerido para que se obtenha um aumento de speedup de 17%?

$$s = \frac{1}{(1-f) + \dfrac{f}{K}} \implies (1-f) + \frac{f}{K} = \frac{1}{s} \implies \ldots$$

$$\implies \frac{f}{K} = \frac{1}{s} - (1-f) \implies \ldots$$

$$K = \frac{f}{\dfrac{1}{s} + f - 1}$$

$$= \frac{0,44}{\dfrac{1}{1,17} + 0,44 - 1}$$

$$= 1.5$$