Relatorio Tecnico - NumerosSPD

SPD - Aluno Gui Costa, n 61172

Objetivo:

Pretende-se determinar os fatores de um número. Tendo em conta o desempenho de diferentes escolhas de implementacao.

```
1. Implementar a fatorizacao sequencialmente.
2. Implementar usando pthreads
3. Iplementar usando OpenMP
4. Implementar usando MPI
5. Analizar e compara o desempenho/resultado dos passos 1-4
```

Para tal, deve-se aplicar os seguintes criterios

```
* 2 se algarismo à direita par
* 3 se soma dos algarismos for múltipla de 3
* 4 se os dois algarismos à direita forem múltiplos de 4
* 5 se o algarismo à direita for 0 ou 5
* 6 se for divisível por 2 e 3
* 9 se for divisível por 3 e soma dos algarismos múltipla de 9
* 10 se for divisível por 2 e 5, i.é, se algarismo à direita for 0
```

Como foram calculados os tempos?

Os templos foram calculados de uma forma simples.
Criou-se um script em bash (apresentado em baixo) para correr o programa para todos os numeros entre 0 e X (por defeito, x = 99999) e mediu-se o tempo total que o programa demorou a executar para todos esses numeros.

```
#!/bin/bash
#filename = loop.sh
a=99999
while [ $a -gt 0 ]; do
        ./SequentialFactor A $a
        let a-=1
done
```

O tempo de exucucao foi medido com o comando" `time source ./loop.sh` ",3 vezes por cada implementacao, obtendo resultados no formato seguinte:

```
real    0m52,494s
user    0m40,536s
sys     0m15,895s
```

> Mais sobre estas metricas/comandos:
> Avaliação de desempenho - SPD UAlg
> What do 'real', 'user' and 'sys' mean in the output of time? - Stack Over

Primeira implementacao - sequencial

```
void factorizedSequential(int x){
    int sum=0,factorSize = 1,digits = x, digitsCount = 0;
    if(x==0) {
        printf("%d\n",0);
        return 0;
    }

    while (digits != 0) {
        digits /= 10;
        digitsCount++;
    }

    int a[digitsCount],factors[12];
    split(a,x,digitsCount);
    factors[0]=1;
    for(int i = 0; i<digitsCount; i++)
        sum+=a[i];

    if(byTwo(a[digitsCount-1])){
        factors[factorSize++]=2;
        if(byFour(a+(digitsCount-2)))
            factors[factorSize++]=4;
        if(byThree(sum)){
            factors[factorSize++]=3;
            factors[factorSize++]=6;
            if(byNine(sum))
                factors[factorSize++]=9;
        }
    } else if(byThree(sum)){
        factors[factorSize++]=3;
        if(byNine(sum))
            factors[factorSize++]=9;
    }
    if(byFive(a[digitsCount-1]))
        factors[factorSize++]=5;
    if(byTen(a[digitsCount-1]))
        factors[factorSize++]=10;

    ints_println_basic(factors,factorSize);
}
```

```
void split(int *a,int x, int dCount){
    if(dCount!=0){
        int count = 0;
        int n = x;

        while (n != 0){
            a[count] = n % 10;
            n /= 10;
            count++;
        }
        int b[dCount];
        n=0;
        for(int i = dCount-1;i>-1;i--)
            b[n++]=a[i];
        for(int i=0;i<dCount;i++)
            a[i]=b[i];
    }
}
int byTwo(int x){
    return x%2 == 0;
}
int byThree(int sum){
    return sum%3==0;
}
int byFour(int *a){
    char num[2];
    num[0] = a[0];
    num[1] = a[1];
    return atoi(num)%4;
}
int byFive(int x){
    return (x==5 || x==0);
}
int byNine(int sum){
    return sum%9==0;
}
int byTen(int x){
    return x==0;
}
```

> O codigo devera esta tambem visivel aqui.

Explicacao

E possivel verificar a implementacao sequencial no codigo a cima.

Esta, simplesmente percorre num unico processo/thread, de maneira sequencial, i.e. instrucao espera que outra acabe para continuar.

Existe a necessidade de separar os digitos do numero (123 => [1,2,3]) de forma que seja possivel aplicar os algoritmos necessarios.Como e o caso da divisao por quatro em que apenas temos de olhar para os ultimos doi digitos do numero.

Resultados

Como descrito a cima foi corrido 3 vezes, por cada vez foram testados numeros de 1 a 99999 (99998 numeros).

```
#Resultados 1
real    0m53,854s
user    0m40,810s
sys     0m15,790s
#Resultados 2
real    0m52,716s
user    0m39,885s
sys     0m15,731s
#resultados 3
real    0m53,177s
user    0m40,727s
sys     0m15,755s
```

A media dos 3 resultados em segundos foi `53.249s`.

Se pegarmos neste media e dividirmos pelo numero de *numeros* testados $53.249s \over 99998 n$ obtemos estimativa igual a $5.325001*10^{-4}$ segundos por numero.

Multiplicando o resultado a cima por 1000 obtemos o valor na casa dos milisegundos.

$$\begin{aligned} 5.325001*10^{-4} * 10^3 = 0.5325001 \end{aligned} $$

milisegundos por numero.