OpenMP

Eng. Software / PSPD

Prof. Fernando W Cruz

Problema a ser paralelizado

Elaborar um programa para encontrar a quantidade de ocorrências do inteiro N em um arquivo de 1 milhão de inteiros (com número fixo de dígitos)

Obs.: Comando usado para criação do arquivo de inteiros (Shell script):

$ for ((i=0;i<1000000;i++)); do printf “%0.8d\n” $RANDOM; done > arqnums.in

Versão serial

```c
#include <stdio.h>
#include <omp.h>
#define MAX 1000000

int main(void) {
    int sumn=0;
    int lido;
    int n=3;

    for (int i=0;i<MAX;i++) {
        scanf("%d", &lido);
        if (lido == n)
            sumn++;
    } /* fim-for */
    printf("Encontramos %d ocorrências de %d\n", sumn, n);
    return 0;
} /* fim programa */
```

```
$ time bin < <arquivo_de_entrada.in>
```

Versão paralela

```c
#include <stdio.h>
#include <omp.h>
#define MAX 1000000

int main(void) {
    int sumn=0;
    int lido;
    int n=3;
    #pragma omp parallel for reduction(+:sumn, lido)
    for (int i=0;i<MAX;i++) {
        scanf("%d", &lido);
        if (lido == n)
            sumn++;
    } /* fim-for */
    printf("Encontramos %d ocorrências de %d\n", sumn, n);
    return 0;
} /* fim programa */
```

```
$ time bin < <arquivo_de_entrada.in>
```

1a. Versão paralela... eficiente?

```c
#include <stdio.h>
#include <omp.h>
#define MAX 1000000

int main(void) {
    int lido, n=3;
    int sumn=0;
    int v[MAX];
    FILE *fd;

    fd=fopen("arqnums.in", "r");
    for (int i=0; i<MAX; i++) {
        fscanf(fd,"%d",&lido);
        v[i] = lido;
    } /* fim-for */
    #pragma omp parallel for reduction(+:sumn)
    for (int i=0; i<MAX; i++)
        if (v[i] == n)
            sumn++;
    printf("Total de %d's = %d\n", n, sumn);

    return 0;
} /* fim-programa */
```


Perceber que o scanf, fscanf são Thread safety (apenas uma execução por vez)

2a. Versão paralela... eficiente???

```c
#include <stdio.h>
#include <omp.h>
#define MAX 1000000

int main(void) {
    int n=3;
    int lido;
    int sumn=0;
    FILE *fd;

    fd=fopen("arqnums.in", "r");
    #pragma omp parallel for reduction(+:sumn, lido)
    for (int i=0; i<MAX; i++) {
        fscanf(fd,"%d",&lido);
        if (lido ==n)
            sumn++;
    } /* fim-for */
    printf("Total de %d's = %d\n", n, sumn);
    return 0;
} /* fim-programa */
```

3a. Versão - separando *scan* da busca

Resolve o problema de desempenho?

```c
#include <stdio.h>
#include <omp.h>
#define MAX 1000000

int main(void) {
    int n=3;
    int lido;
    int sumn=0;
    int v[MAX];
    FILE *fd;

    fd=fopen("arqnums.in", "r");
    for (int i=0; i<MAX; i++) {
        fscanf(fd,"%d",&lido);
        v[i] = lido;
    } /* fim-for */
    #pragma omp parallel for reduction(+:sumn)
    for (int i=0; i<MAX; i++)
        if (v[i] == n)
            sumn++;
    printf("Total de %d's = %d\n", n, sumn);

    return 0;
} /* fim-programa */
```

Não absolutamente, porque o scan continua sendo um gargalo p a performance

Dica

Dividir o arquivo proporcional ao número de threads e fazer com que cada thread verifique as ocorrências de N em uma determinada parte desse arquivo.

Cuidado: calcular o offset para cada thread

Possível solução para o problema

```c
#include <stdio.h>
#include <omp.h>
#define MAX 1000000

int main(void) {
    int n=3, larg=9;
    int lido;
    int sumn=0;
    int offset;
    FILE *fd;

    #pragma omp parallel private(lido, offset) reduction(+:sumn)
    {
        FILE *fd=fopen("arqnums.in", "r");
        offset = (MAX/omp_get_num_threads())*larg*omp_get_thread_num();
        fseek(fd, offset, SEEK_SET);
        #pragma omp for
        for (int i=0; i<(MAX/omp_get_num_threads());i++) {
            fscanf(fd, "%d", &lido);
            if (lido == n)
                sumn++;
        } /* fim-for */
    }
    printf("Total de %d's = %d\n", n, sumn);
    return 0;
} /* fim-programa */
```

O desempenho é melhor, porque cada arquivo olha uma região distinta do arquivo

OpenMP

Eng. Software / PSPD

Prof. Fernando W Cruz