Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Programação Concorrente

Relatório de Entrega de Atividades

Aluno(s): Henrique Mendes de Freitas Mariano; Leonardo Rodrigues de Souza
Matrícula: 170012280; 170060543
Atividade: Aula Prática 07 - OpenMP

1.1 - Tempo Base (Time)

Média = (1 + 0 + 1) / 3 = 0,666666667

```c
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 1-1-tempo-base.c
// atividade: 1.1
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>
// #include <omp.h>

long long int somavalores(int *valores, int n) {
    long long int soma = 0;

    for (int i = 0; i < n; i++) {
        soma = soma + valores[i];
    }

    return soma;
}

int main() {
    long long int i, n, soma;
    time_t inicio, fim;
    int *valores;
    // scanf("%lld", &n);
```

```
    n = 100000000;

    valores = (int *) malloc(n * sizeof(int));
    for (i = 0; i < n; i++) {
        valores[i] = 1;
    }

    inicio = time(NULL);
    soma = somavalores(valores, n);
    fim = time(NULL);

    printf("Soma: %lld - %s - wall time: %f\n", soma, soma == n ? "ok" :
"falhou", difftime(fim, inicio));

    free(valores);
    valores = NULL;

    return 0;
}
```

2.1 - Tempo Base (omp)

Média = (0.230003 + 0.228391 + 0.228496) / 3 = 0.228963333

```
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 1-2-tempo-base-omp.c
// atividade: 1.2

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

long long int somavalores(int *valores, int n) {
    long long int soma = 0;
    int i;
    for (i = 0; i < n; i++) {
        soma = soma + valores[i];
    }
```

```
    return soma;
}

int main() {
    long long int i, n, soma;
    double inicio = 0, fim = 0;
    int *valores;
    // scanf("%lld", &n);
    n = 100000000;
    valores = (int *) malloc(n * sizeof(int));

    for (i = 0; i < n; i++) {
        valores[i] = 1;
    }

    inicio = omp_get_wtime();
    soma = somavalores(valores, n);
    fim = omp_get_wtime();

    printf("Soma: %lld - %s - wall time: %lf\n", soma, soma == n ? "ok" :
"falhou", (fim - inicio));

    free(valores);
    valores = NULL;

    return 0;
}
```

1.3 -

1 Thread -> Média = (1.448061 + 1.471480 + 1.439597) / 3 = 1.453046

2 Threads -> Média = (2.377274 + 2.777756 + 3.010234) / 3 = 2.721754667

4 Threads -> Média = (4.106781 + 4.158676 + 4.101935) / 3 = 4.122464

```
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 1-3-soma-pararell.c
```

```c
// atividade: 1.3
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

long long int somavalores(int *valores, int n) {
    long long int soma = 0;
    omp_set_num_threads(4);
    #pragma omp parallel
    {
        for (int i = 0; i < (n/4); i++) {
            #pragma omp critical
                soma = soma + valores[i];
        }
    }
    return soma;
}

int main() {
    long long int i, n, soma;
    double inicio = 0, fim = 0;
    int *valores;
    // scanf("%lld", &n);
    n = 100000000;
    valores = (int *) malloc(n * sizeof(int));

    for (i = 0; i < n; i++) {
        valores[i] = 1;
    }
    {
        inicio = omp_get_wtime();
        soma = somavalores(valores, n);
        fim = omp_get_wtime();
    }

    printf("Soma: %lld - %s - wall time: %lf\n", soma, soma == n ? "ok" :
"falhou", (fim - inicio));
```

```
    free(valores);
    valores = NULL;

    return 0;
}
```

1.4 -

1 Thread -> Média = (0.720577 + 0.720578 + 0.721332) / 3 = 0.720829

Diferença de 0.732217 segundos

2 Threads -> Média = (1.510746 + 3.020549 + 1.633403) / 3 = 2.054899333

Diferença de 0.666855334 segundos

4 Threads -> Média = (2.415259 + 2.182536 + 2.452326) / 3 = 2.350040333

Diferença de 1.772423667 segundos

1.5 -

1 Thread -> Média = (0.229605 + 0.230283 + 0.228296) / 3 = 0.229394667

2 Threads -> Média = (0.099134 + 0.099926 + 0.098677) / 3 = 0.099245667

4 Threads -> Média = (0.085650 + 0.087738 + 0.084263) / 3 = 0.085883667

```
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 1-5-soma-parcial.c
// atividade: 1.5
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

long long int somavalores(int *valores, int n) {
    long long int soma = 0;
    long long int soma_parcial = 0;
    omp_set_num_threads(4);
    #pragma omp parallel private(soma_parcial)
    {
        for (int i = 0; i < (n/4); i++) {
            soma_parcial = soma_parcial + valores[i];
```

```
        }

        #pragma omp atomic
            soma = soma + soma_parcial;
    }
    return soma;
}

int main() {
    long long int i, n, soma;
    double inicio = 0, fim = 0;
    int *valores;
    // scanf("%lld", &n);
    n = 100000000;
    valores = (int *) malloc(n * sizeof(int));

    for (i = 0; i < n; i++) {
        valores[i] = 1;
    }
    {
        inicio = omp_get_wtime();
        soma = somavalores(valores, n);
        fim = omp_get_wtime();
    }

    printf("Soma: %lld - %s - wall time: %lf\n", soma, soma == n ? "ok" :
"falhou", (fim - inicio));

    free(valores);
    valores = NULL;

    return 0;
}
```

1.6 -

```c
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 1-6-omp-for.c
// atividade: 1.6
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

long long int somavalores(int *valores, int n) {
    long long int soma = 0;
    long long int soma_parcial = 0;
    omp_set_num_threads(4);
    #pragma omp parallel for
        for (int i = 0; i < n; i++) {
            #pragma omp atomic
                soma = soma + valores[i];
        }

    return soma;
}

int main() {
    long long int i, n, soma;
    double inicio = 0, fim = 0;
    int *valores;
    // scanf("%lld", &n);
    n = 100000000;
    valores = (int *) malloc(n * sizeof(int));

    for (i = 0; i < n; i++) {
        valores[i] = 1;
    }
    {
        inicio = omp_get_wtime();
        soma = somavalores(valores, n);
        fim = omp_get_wtime();
```

```
    }

    printf("Soma: %lld - %s - wall time: %lf\n", soma, soma == n ? "ok" :
"falhou", (fim - inicio));

    free(valores);
    valores = NULL;

    return 0;
}
```

1.7 - Redução

```
// autores: Henrique Mendes de Freitas Mariano e Leonardo Rodrigues de Souza
// arquivo: 1-7-reducao.c
// atividade: 1.7
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

long long int somavalores(int *valores, int n) {
    long long int soma = 0;
    long long int soma_parcial = 0;
    omp_set_num_threads(4);
    #pragma omp parallel for reduction(+: soma)
        for (int i = 0; i < n; i++) {
            #pragma omp atomic
                soma = soma + valores[i];
        }

    return soma;
}

int main() {
    long long int i, n, soma;
    double inicio = 0, fim = 0;
```

```
    int *valores;
    // scanf("%lld", &n);
    n = 100000000;
    valores = (int *) malloc(n * sizeof(int));

    for (i = 0; i < n; i++) {
        valores[i] = 1;
    }
    {
        inicio = omp_get_wtime();
        soma = somavalores(valores, n);
        fim = omp_get_wtime();
    }

    printf("Soma: %lld - %s - wall time: %lf\n", soma, soma == n ? "ok" :
"falhou", (fim - inicio));

    free(valores);
    valores = NULL;

    return 0;
}
```