

Bacharelado em Ciência da Computação - DCC/IM-UFRJ
Programação Paralela e Distribuída
Prof. Gabriel P. Silva
3ª Lista de Exercícios – 2/12/2008

1. Para quais tipos de arquitetura se destina o padrão OpenMP?
2. Explique o funcionamento das regiões paralelas no OpenMP?
3. Que valor é retornado pela rotina `omp_get_num_threads()`?
4. Qual cláusula é utilizada para dar valor inicial a uma variável local?
5. Dado um laço paralelizado, quais variáveis são consideradas locais como padrão?
6. Enumere e descreva quatro tipos de operações de redução utilizadas na diretiva `reduces` do OpenMP?
7. Explique as diferenças entre os escalonamentos `STATIC`, `DYNAMIC`, `GUIDED` ou `RUNTIME`.
8. No código abaixo, faça um diagrama do escalonamento das iterações do laço entre 4 threads.

```
#pragma omp for schedule(guided, 5)
  for (j = 0; j < 50; j++){
      a[j]= b[j] + c[j];
  }
```

9. Reescreva o código seguinte, colocando as operações de barreira explicitamente, de modo a maximizar o desempenho, mas sem alterar o resultado da execução.

```
#pragma omp parallel
{
  #pragma omp for
  for (j = 0; j < n; j++){
      a[j]= b[j] + c[j];
  }
  #pragma omp for
  for (j = 0; j < n; j++){
      d[j] = e[j] * f;
  }
  #pragma omp for
  for (j = 0; j < n; j++){
      z[j] = (a[j]+a[j+1]) * 0.5;
  }
}
```

10. Paralelize o exemplo a seguir **sem** o uso da diretiva `parallel for`.

```
for (i = 0; i < n; i++)
{
    z[i] = a * x[i] + y;
}
```

11. Generalize o exemplo a seguir para um número arbitrário de threads. Considere que o número de colunas é maior que o número de threads em execução.

```
#pragma omp parallel default(none) private (i, myid) shared(a,n)

myid = omp_get_thread_num();

for(i = 0; i < n; i++)
{
    a[i][myid] = 1.0;
}
```

12. Explique porque cada um dos laços a seguir pode ou não pode ser paralelizado com a diretiva `parallel for`.

- a)

```
for (i = 0; i < N ; i ++)  
{  
    if (x [ i ] > maxval) break;  
}
```
- b)

```
for (i = 0; i < N ; i ++)  
    for (j = 0; j < i ; j++)  
    {  
        a[ j ][ i ] = a[ j ][ i ] + a[ j+1, i ];  
    }
```
- c)

```
for (i = 0; i < N ; i ++)  
    if (weight [ i ] . HEAVY) {  
        pid = fork ( );  
        if (pid == -1 ) {  
            perror("fork");  
            exit (1);  
        }  
        if (pid == 0 ) {  
            heavy_task( );  
            exit (1);  
        }  
        else  
            light_task ( );  
    }  
}
```
- d)

```
for (i = 0; i < N ; i ++)  
{  
    a [ i ] = a [ i ] * a [ i ];  
    if (fabs(a[ i ]) > machine_max ||  
        fabs(a[ i ]) < machine_min)  
    {  
        printf ("i = %d \n", i);  
        break;  
    }  
}
```

13. Considerando os seus conhecimentos de cache, que considerações você faria sobre a influência do tamanho do "chunk" no escalonamento de um laço `for` sobre o desempenho da execução em um sistema multiprocessador?

14. Escreva uma rotina para calcular o produto escalar de dois vetores utilizando rotinas do OpenMP. Considere cada vetor com 1000 posições. Utilize a cláusula de redução.

15. Considere o seguinte laço:

```
x= 1;
```

```
#pragma omp parallel for firstprivate (x)
```

```
for(i = 0; i < n; i++){  
    y [ i ] = x + i;  
    x = i;  
}
```

- a) Porque este laço está incorreto? y [i] recebe o mesmo resultado independente do número de threads executando o laço?
- b) Qual o valor de *i* ao final do laço? Qual é o valor de x ao final do laço?
- c) Qual seria o valor de x ao final do laço se o seu escopo fosse *shared*?
- d) Este laço pode ser paralelizado corretamente (isto é, preservando a semântica seqüencial) apenas com o uso de diretivas?