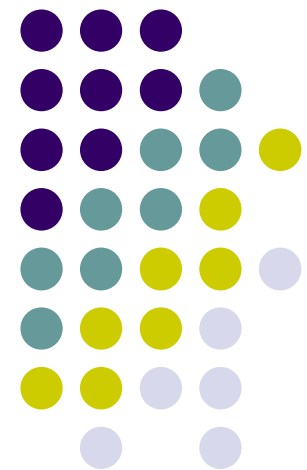


Computação Paralela

OpenMP

João Luís Ferreira Sobral
Departamento de Informática
Universidade do Minho

Novembro 2006





OpenMP

Standard para programação de sistemas de memória partilhada

- Promovido pela Intel para programação de processadores com *hyperthreading*
- Implementado pelo compilador C++ 7.0 da Intel
- Será suportado na versão de gcc 4.0
- Será uma referência para os processadores *multicore*?
- Proporciona uma alternativa (mais simples) do que a utilização de *threads* em Java para desenvolvimento de aplicações
- Baseada em directivas que são ignoradas pelos compiladores que não suportam o standard:

```
# pragma omp parallel for
    for(int i=0; i<9; i++) {
        printf(“%d”,i);
    }
```



OpenMP

Paradigma base

- Baseado na especificação de ciclos ou secções de código cuja execução pode ser executada em paralelo
- Potencialmente cada iteração do ciclo pode ser executada por um fio de execução diferente
- Os detalhes da criação e destruição de fios são geridos pelo compilador
- O número adequado de fios de execução é determinado pela biblioteca do OpenMP em função dos recursos de hardware disponíveis
- Exemplo – conversão de uma imagem a cores para tons de cinzento:

```
# pragma omp parallel for
    for(int i=0; i<numPixels; i++) {
        grayScale[i] = 0.299*rgb[i].red + 0.587*rgb[i].green + 0.114*rgb[i].blue;
    }
```



OpenMP

Desafios na paralelização de ciclos

1. Dependências entre iterações do ciclo (obrigam à reescrita do ciclo):

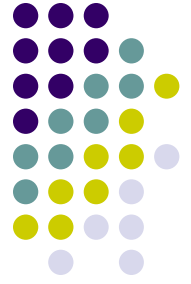
```
for(int i = 2; i<10; i++) {  
    fact[i] = i * fact[i-1];  
}
```

2. Corridas entre fios de execução (obrigam à introdução de sincronização)

```
for(int i = 0; i<6; i++) {  
    num = rand() +1;  
    if (num>60) num = num % 60 +1;  
}
```

- Variáveis privadas a cada iteração do ciclo:
 - Declarar a variável dentro do ciclo:
 - int num;
 - Utilizar a primitiva private
 - # pragma omp parallel for *private(num)*

OpenMP



Desafios na paralelização de ciclos (cont.)

- Escalonamento e balanceamento dos ciclos
 - As iterações do ciclo podem não demorar todas o mesmo tempo a executar
 - Por defeito o escalonamento das iterações é estático (í.e., assume que todas as iterações demoram o mesmo tempo a executar)
 - Tipos de escalonamento

`# pragma omp parallel for schedule(kind [,chunk size])`

- Estático: o número de iterações é dividido pelo número de processadores (*chunk size*)
- Dinâmico: a várias iterações são colocadas numa fila de trabalho e cada fio de execução retira uma tarefa da fila de cada vez (ou o número especificado por *chunk size*)
- Guiado: semelhante ao dinâmico mas o *chunk size* vai diminuindo durante a execução



OpenMP

Desafios na paralelização de ciclos (cont.)

- Redução de vectores de valores a um só valor:

```
sum = 0;  
# pragma omp parallel for reduction(+:sum)  
  for(int i = 0; i<100; i++) {  
    sum += array[i];  
  }
```
- Reutilização de fios de execução entre vários ciclos

```
# pragma omp parallel {  
  #pragma omp for  
    for(int i = 0; i<100; i++)  
      ...  
  #pragma omp for  
    for(int j= 0; j<100; j++)  
      ...  
}
```

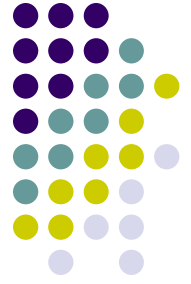


OpenMP

Secções de código com execução em paralelo

- Permitem a especificação de tarefas heterogéneas

```
#pragma omp parallel {  
  #pragma omp sections {  
    #pragma omp section {  
      taskA();  
    }  
    #pragma omp section {  
      taskB();  
    }  
    #pragma omp section {  
      taskC();  
    }  
  }  
}
```



OpenMP

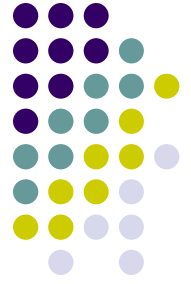
Primitivas de sincronização

- Remover a barreira no fim do ciclo for

```
# pragma omp parallel {  
#pragma omp for nowait  
    for(int i = 0; i<100; i++)  
        ...  
#pragma omp for  
    for(int j= 0; j<100; j++)  
        ...  
}
```

- Acrescentar uma barreira

```
# pragma omp parallel {  
    ...  
#pragma omp barrier  
    ...  
}
```

OpenMP

Primitivas de sincronização (cont.)

- Secções críticas (executadas com exclusão mútua)
pragma omp *critical*
if (max < newVal) max = newVal;
- Operações atómicas
pragma omp *atomic*
a[i] +=x;
- Especificar uma operação a ser executada por um só fio de execução: **omp *single***



OpenMP

Exercícios (compilar no Search3 com `icc -openmp fich`)

- Alterar o seguinte programa para efectuar os *printf* em paralelo

```
for(int i=0; i<1000; i++) {  
    printf("Olá %d ",i);  
}
```

- Calcular o valor Pi em paralelo:

```
double f( double a ) { return (4.0 / (1.0 + a*a)); }  
  
double pi = 3.141592653589793238462643;
```

$$\pi = \int_0^1 \frac{4}{1+x^2}$$

```
int main() {  
    double mypi = 0;  
    int n = 10000000; // número de pontos do integral a calcular  
    float h = 1.0 / n;  
  
    for(int i=0; i<n; i++) {  
        mypi = mypi + f(i*h);  
    }  
    mypi = mypi * h;  
  
    printf("Aproximacao de pi = %.10f \n", (pi - mypi));  
}
```