

Programação Concorrente e Paralela

OpenMP

Noemi Rodriguez

2016



o que é

- padrão para programação com threads e memória compartilhada em aplicações científicas
- idéia de paralelização incremental (e almoço gratis ☺!?!?)

- www.openmp.org
- implementado no gcc a partir de 4.3.2



como é

- diretivas (pragmas) em C/C++
 - comentários em Fortran
- e mais uma biblioteca



- diretiva se aplica a comando a seguir:

```
comando 1;  
#pragma <diretiva OpenMP>  
comando 2;  
comando 3;
```

- ênfase é na paralelização de ciclos:

```
void main() {  
    ...  
    #pragma omp parallel for  
    for (int i=0; i<1000; i++) {  
        a[i] =  
            ...  
    }  
    ...  
}
```



Uma comparação simples:

```
Dot-Product in OpenMP
int main(argc,argv)
int argc;
char *argv[];
{
double sum;
double a [256], b [256];
int n=256;
for (i = 0; i < n; i++) {
a [i] = i * 0.5;
b [i] = i * 2.0;
}
sum = 0;
#pragma omp for reduction(+:sum)
for (i = 1; i <= n; i++) {
sum = sum + a[i]*b[i];
}
printf ("sum = %f \n", sum);
}
```

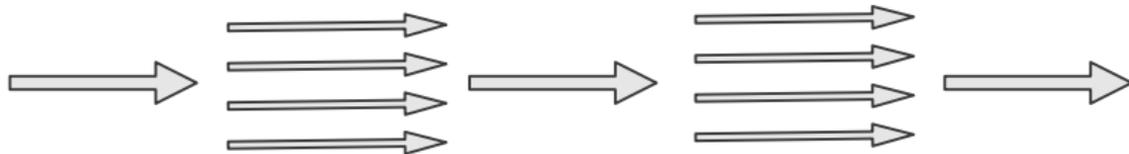
```
Dot-Product in Pthreads
#define NUMTDS 4
double sum;
double a [256], b [256];
int status;
int n=256;
pthread_t thd[NUMTDS];
pthread_mutex_t mutexsum;
int main(argc,argv)
int argc;
char *argv[];
{
pthread_attr_t attr;
for (i = 0; i < n; i++) {
a [i] = i * 0.5;
b [i] = i * 2.0;
}
pthread_mutex_init(&mutexsum, NULL);
pthread_attr_init(&attr);
pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
for (l=0; l<NUMTDS; l++)
{
pthread_create( &thds[l], &attr, dotprod, (void *)l);
}
pthread_attr_destroy(&attr);
for (l=0; l<NUMTDS; l++) {
pthread_join( thds[l], (void **)&status);
}
printf ("sum = %f \n", sum);
pthread_mutex_destroy(&mutexsum);
pthread_exit(NULL);
}
void *dotprod(void *arg)
{
int myid, i, my_first, my_last;
double sum_local;
myid = (int)arg;
my_first = myid * n/NUMTDS;
my_last = (myid + 1) * n/NUMTDS;
sum_local = 0;
for (i = my_first; i <= my_last; i++) {
sum_local = sum_local + a [i]*b[i];
}
pthread_mutex_lock (&mutex_sum);
sum = sum + sum_local;
pthread_mutex_unlock (&mutex_sum);
pthread_exit((void*) 0);
}
```



- não há garantias sobre correção do programa a ser executado
- programador deve controlar com diretivas e através do próprio programa



Fluxo de um programa openMP



- não há garantias sobre correção do programa a ser executado
- programador deve controlar com diretivas e através do próprio programa
- diretivas: paralelização, dados, sincronização



Diretivas de paralelização

- `#pragma omp parallel`
 - comando (estruturado) a seguir executado em paralelo por `n` threads
- total ênfase em SPMD
- threads podem seguir caminhos diferentes usando seus IDs
 - `int omp_get_thread_num()`
 - `int omp_get_num_threads()`



Diretivas de paralelização

- diretiva mais frequente: parallel for

```
#pragma omp parallel  
#pragma omp for  
for( ... ) { ... }
```

- iterações devem ser independentes
- for deve ter formato limitado
for (index-start; index oprel end; index op= index)
- break, return, exit, goto não são permitidos
- variável de controle é a única privada por default



Diretivas de paralelização

- o padrão frequente:

```
#pragma omp parallel  
#pragma omp for  
for ( ; ; ) { ... }
```

- pode ser reescrito como:

```
#pragma omp parallel for  
for ( ; ; ) { ... }
```

- cuidados com a semântica fork/join!



- paralelismo condicional
- controle do escalonamento



- variáveis podem ser “declaradas” como `private` (uma cópia por thread)
- variáveis de *redução*

redução

- uso frequente em aplicações de computação científica!



Redução



- algumas operações pré-definidas

```
#pragma omp parallel for reduction( +:sum )  
for( i=0; i<n; i++ )  
    sum += a[i];
```



Diretivas de sincronização

```
#pragma omp critical name
```

```
#pragma omp critical barrier
```

- não há suporte para espera por condição!



- simplicidade para obter algum benefício de paralelização em aplicações com estrutura “embaraçosamente paralela”
- obtenção de bom desempenho pode se tornar tarefa complicada
 - remover ou reduzir dependências em loops
 - transformar limites de loops em fronteiras fixas
 - implementar espera por condição com espera ocupada
 - ...
 - diretivas sofisticadas emergindo em algumas implementações

