Programação Concorrente e Paralela

OpenMP

Noemi Rodriguez

2016

pthreads
padrão posix

OpenMP

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

OpenMP

como é

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

OpenMP

- 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 status;
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);
}
```

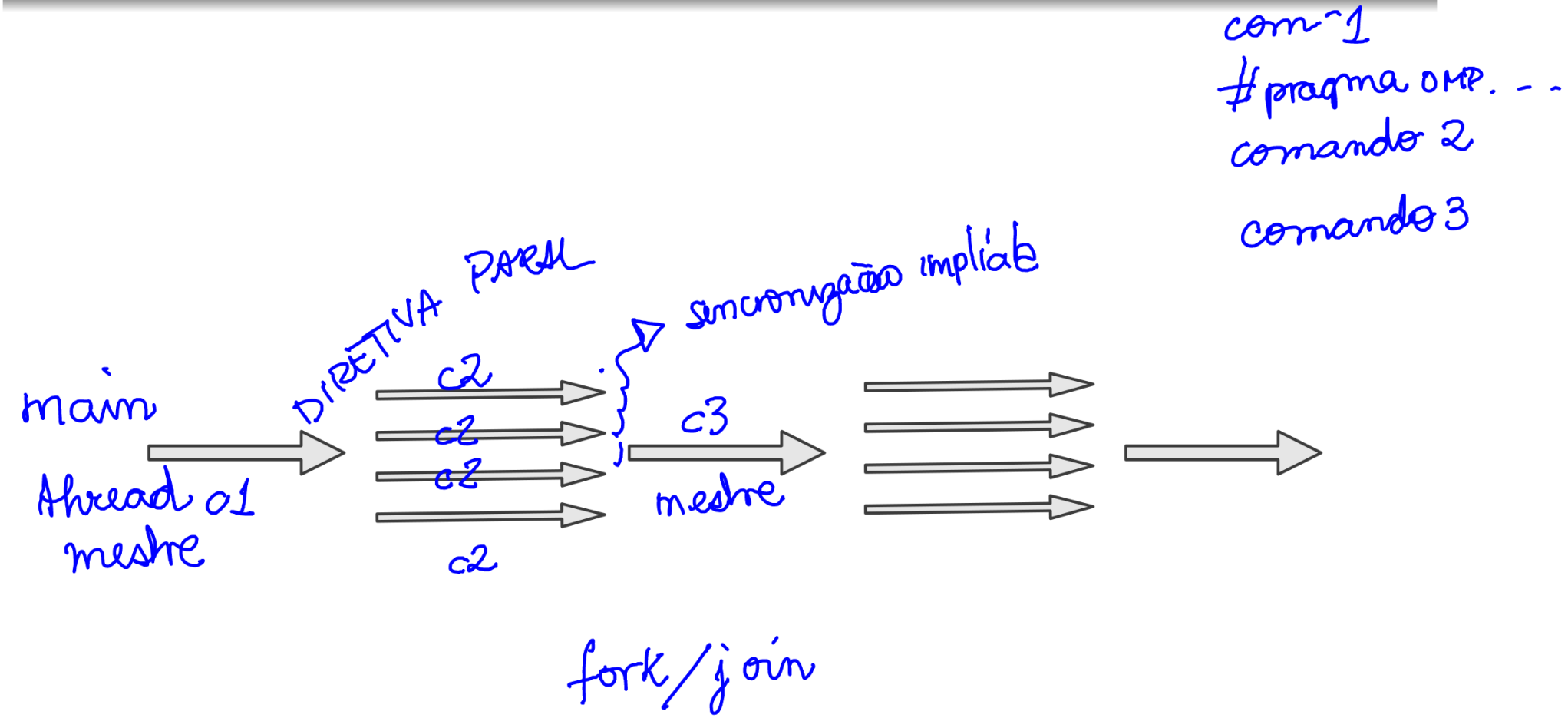
Dot-Product in Pthreads

```
#define NUMTHRDS 4
double sum;
double a [256], b [256];
int status;
int n=256;
pthread_t thd[NUMTHRDS];
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;
}
thread_mutex_init(&mutexsum, NULL);
pthread_attr_init(&attr);
pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
for(i=0;i<NUMTHRDS;i++)
{
pthread_create( &thds[i], &attr, dotprod, (void *)i);
}
pthread_attr_destroy(&attr);
for(i=0;i<NUMTHRDS;i++) {
pthread_join( thds[i], (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/NUMTHRDS;
my_last = (myid + 1) * n/NUMTHRDS;
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);
}
```

Observações

- 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



Observações

- 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 SINGLE PROGRAM MULTIPLE DATA
- 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

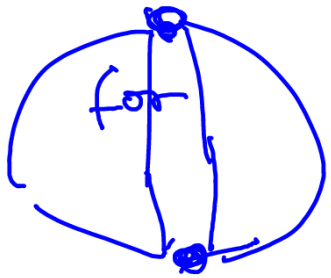
Diretivas de paralelização

- 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!

```
#pragma omp parallel
{printf("abc\n");for(i=0;~)
 #pragma omp for
 for( ~ ...){
    /*for 1*/
 }#pragma OMP for
 for(~~){
    /*for 2*/
 }
}
```

```
#pragma omp parallel for
for(~...){
   for 1
}
#pragma omp parallel for
for(~...){
  for 2
}
```

Observações

- paralelismo condicional
- controle do escalonamento

Diretivas de dados

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

redução

- uso frequente em aplicações de computação científica!

```
sum = 0;
for (i=0; i<LIM; i++)
    sum += a[i];
```

array de acumuladores, 1 por thread

→ `sum[myid] += a[i]`

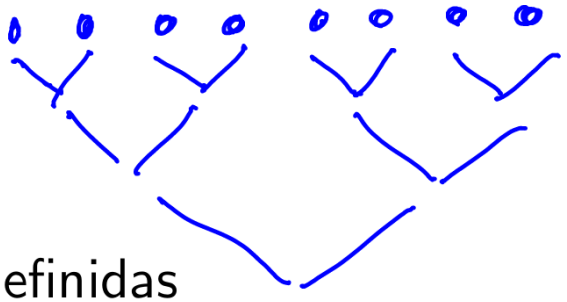
depois do loop

```
#omp critical
gsum += sum[myid];
```

Redução

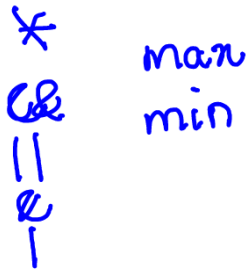
apoio embutido a operações que tomam 1 resultado parcial de cada thread e retornam 1 resultado global

Diretivas de 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 nome SOMA

#pragma omp critical barrier
```

- não há suporte para espera por condição!

OpenMP

- 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 fizas
  - implementar espera por condição com espera ocupada
  - ...
  - diretivas sofisticadas emergindo em algumas implementações