OpenMP

Laborator 1

Arhitecturi si Prelucrari Paralele

Sumar

Concepte de paralelizare

Introducere in OpenMP

Concepte de paralelizare

Modele de arhitecturi paralele: SISD, SIMD, MISD, MIMD
Dependenta de date
Sincronizari

Implementari:
- Thread-uri sau procese
- Paralel sau distribuit

Introducere in OpenMP

Exercitiu (5min):

Fie un sir de numere intregi, initializat. Se cere suma numerelor din sir. Ganditi un algoritm paralel. Distribuit?

Introducere in OpenMP

- Framework pentru shared memory programming
  - directive de compilator (ex: #pragma omp parallel)
  - biblioteca de sistem (ex: omp_get_num_threads(), -lgomp, omp.h)
- Avantaje
  - standardizare si portabilitate – ultimul standard OpenMP 5.0 (www.openmp.org)
  - usurinta in utilizare (vs MPI) – se rescrie doar sectiunea ce merge paralelizata
- Bazat pe Thread-uri
- Foloseste modelul Fork-Join

Introducere in OpenMP

- Regiuni paralele

```
#pragma omp parallel
<code_block>

#pragma omp parallel
{
    printf("hello world\n");
}
```

- Functii
  - **omp_get_num_threads()** – returneaza nr. de thread-uri
  - **omp_set_num_threads()** – seteaza nr . de thread-uri
  - **omp_get_thread_num()** – returneaza thread id-ul curent
  - **omp_get_num_procs()** – returneaza nr. de procesoare disponibile
- https://computing.llnl.gov/tutorials/openMP/#RunTimeLibrary

Introducere in OpenMP

Rularea unui program OpenMP

gcc –fopenmp –o hello hello.c

export OMP_NUM_THREADS=4

./hello

Introducere in OpenMP

Exercitiu (5min):

Scrieti in C un program ce afiseaza “Hello world” in paralel. Variati numarul de thread-uri folosind variabila de mediu OMP_NUM_THREADS. Puteti folosi codul din ex1.c si ex1-1.c.

Introducere in OpenMP

Vizibilitatea variabilelor

Shared – accesibila tuturor thread-urilor

Private – locala thread-ului

Introducere in OpenMP

```
int a, b, c;
a = 2, b = 3, c = 4;
#pragma omp parallel private(a, b) shared(c)
{
   a = 1, b = 2;
   c = a + b;
}
printf("%d %d %d\n", a, b, c);
```

Introducere in OpenMP

Exercitiu (5min):

Scrieti un program in C ce realizeaza suma unui vector de numere intregi:

- Vectorul va avea cel putin 10Mil elemente
- Contorizat timpul de efectuare a adunarii

Puteti folosi codul de la ex2.c.

Pasul 1

```
t1 = omp_get_wtime();
for (i=0;i<N;i++)
   sum += a[i];
t2 = omp_get_wtime();
double duration = t2-t1;
```

```
int sum = 0;
t1 = omp_get_wtime();
#pragma omp parallel default(shared) private(i)
{
for (i=0;i<N;i++)
   sum += a[i];
}
t2 = omp_get_wtime();
double duration = t2-t1;
```

Pasul 2

```
t1 = omp_get_wtime();
#pragma omp parallel default(shared) private(i,tid)
{
tid = omp_get_thread_num();
numt = omp_get_num_threads();

int from, to;
from = (N/numt) * tid;
to = (N/numt)*(tid+1)-1;

if (tid == numt-1)
   to = N-1;

for (i=from;i<=to;i++)
  sum += a[i];
}
t2 = omp_get_wtime();
double duration = t2-t1;
```

```
t1 = omp_get_wtime();
#pragma omp parallel default(shared) private(i,tid)
{
tid = omp_get_thread_num();
numt = omp_get_num_threads();

int from, to;
from = (N/numt) * tid;
to = (N/numt)*(tid+1)-1;

if (tid == numt-1)
  to = N-1;

for (i=from;i<=to;i++)
  #pragma omp critical
  sum += a[i];
}
t2 = omp_get_wtime();
double duration = t2-t1;
```

Introducere in OpenMP

Sincronizari:

Critical - *#pragma omp critical*

- Se executa pe rand de fiecare thread (serializare?)

Barrier - *#pragma omp barrier*

- Se asteapta terminarea thread-urilor din bloc

Nowait - *#pragma omp nowait*

- Thread-urile pot fi refolosite la task-uri ulterioare (for dupa for)

Introducere in OpenMP

Bucle DO/For

```
#pragma omp for
for (i=0;i<N;i++)
  sum += a[i]
```

Introducere in OpenMP

Exercitiu 3 (10min):

Sa se calculeze (corect) suma unui vector folosind calculul paralel oferit de openmp.