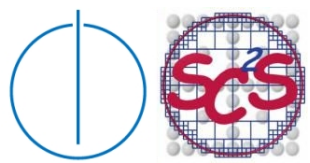
PSE Molekulardynamik

OpenMP

Wolfgang Eckhardt
Thomas Auckenthaler

Technische Universität München, Germany



```
#include <omp.h>
...
#pragma omp parallel for
for(i = 0; i < n; i++)
{
   for(j = 0; j < n; j++)
   {
      do_some_work(i, j);
   }
}
```

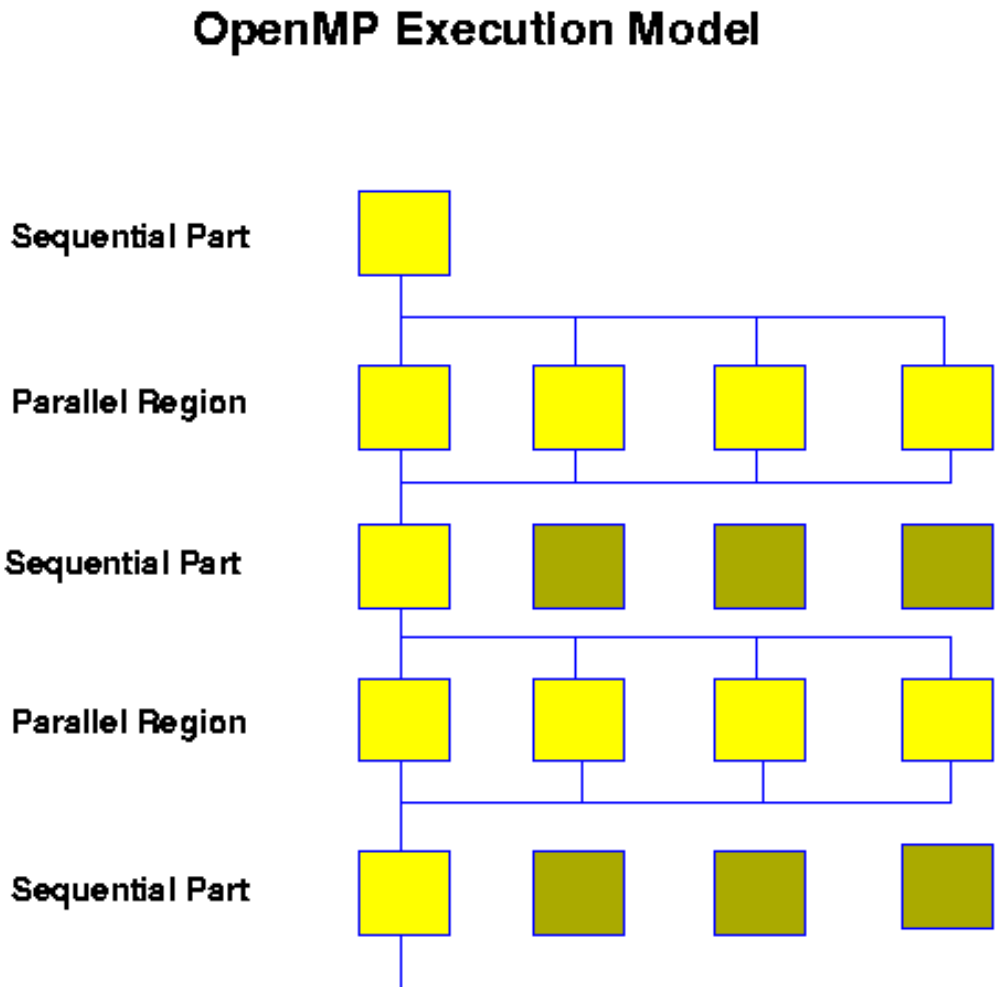


Abbildung: Fork-Join Parallelism.

Überblick

- Compiler-Direktiven
  `#pragma omp ... [clause[[,] clause]...]`
  - Work-sharing
    `#pragma omp for ...`, u.a.
  - Synchronisation
    `#pragma omp barrier`, u.a.
- Data Scope Clauses
  `shared`, `private`, `firstprivate`, `reduction`, u.a.
- Bibliotheksroutinen
  `omp_get_num_threads()`,
  `omp_get_thread_num()`, u.a.

Compilieren und ausführen

- compilieren mit zusätzlichem Compiler-Flag `-openmp` / `-fopenmp`:

```
icc -openmp -o my_alg my_alg.c
gcc -fopenmp -o my_alg my_alg.c
```

- ausführen:

```
export OMP_NUM_THREADS=number_of_threads
```
(default = 1)

```
./my_alg
```

Parallel Region Construct, OpenMP API

```
#pragma omp parallel [clause[[,] clause]...]
structured block
```

- Anweisungen innerhalb einer `parallel region` werden von **allen** Threads ausgeführt

Beispiel

```
#include <omp.h>
...
#pragma omp parallel private(size, rank)
{
   size = omp_get_num_threads();
   rank = omp_get_thread_num();
   printf("Hello World! (Thread %d of %d)", rank,
size);
}
```

OpenMP data scope clauses

- `private`(*list*): deklariert die Variablen in *list* als private für jeden Thread.
- `shared`(*list*): Variablen aus *list* werden von allen Threads gemeinsam genutzt.
- `firstprivate`(*list*): private Variablen, die mit dem letztem Wert vor dem parallelen Abschnitt initialisiert werden.
- `lastprivate`(*list*): nach dem Verlassen des parallelen Abschnitts übernimmt die Variable den Wert von dem Thread, der die letzte Schleifeniteration ausgeführt hat.
- default data scope clause ist `shared`, mit Ausnahmen...
  - lokale Variablen sind `private`
  - Schleifen-Variablen der parallelisierten `for`-Schleifen sind `private`

synchronization constructs

- `#pragma omp barrier`
  - blockiert, bis alle Threads die barrier erreicht haben
- ...

Bibliotheksroutinen

- `omp_get_num_threads()`
  liefert die Anzahl der aktiven Threads
- `omp_get_thread_num()`
  liefert die jeweilige Thread-ID
- `omp_get_wtime()`
  Zeitmessung

References:

- `https://computing.llnl.gov/tutorials/openMP/`
- `http://openmp.org`