



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Praktikum: Paralleles Programmieren für Geowissenschaftler

Prof. Thomas Ludwig, Hermann Lenhart, Felix Hoffmann



Dr. Hermann-J. Lenhart

hermann.lenhart@zmaw.de



OpenMP – Allgemeine Einführung I

OpenMP Merkmale:

OpenMP ist keine Programmiersprache!

OpenMP Notationen werden zu einem sequentiellen FORTRAN Programmen hinzugefügt um anzugeben wie die Arbeit auf die Prozesse verteilt wird und wie der gemeinsame Speicherzugriff erfolgen soll.

Wichtigstes Kriterium für OpenMP ist die Nutzung eines gemeinsamen Speichers.
(shared memory application)

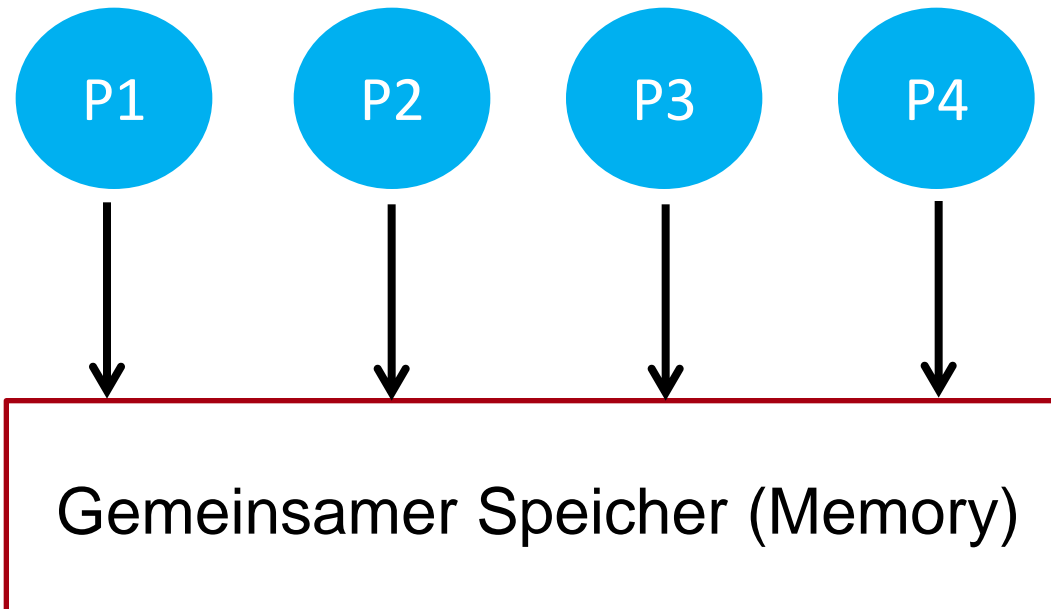


OpenMP – Allgemeine Einführung II

OpenMP Merkmal gemeinsamer Speicher:

SMP:

Symmetrisches Multiprozessersystem
(symmetric multiprocessing)





OpenMP – Allgemeine Einführung III

OpenMP inkrementeller Einsatz in sequentiellm FORTRAN Programm:

Subroutine saxpy (z,a,x,y,n)

Integer :: i, n

Real :: a, y, z(n), x(n)

do i = 1,n

z(i) = a + x(i)+y

enddo

return

end

Subroutine saxpy (z,a,x,y,n)

Integer :: i, n

Real :: a, y, z(n), x(n)

!\$omp parallel do

do i = 1,n

z(i) = a + x(i)+y

enddo

!\$omp end parallel do

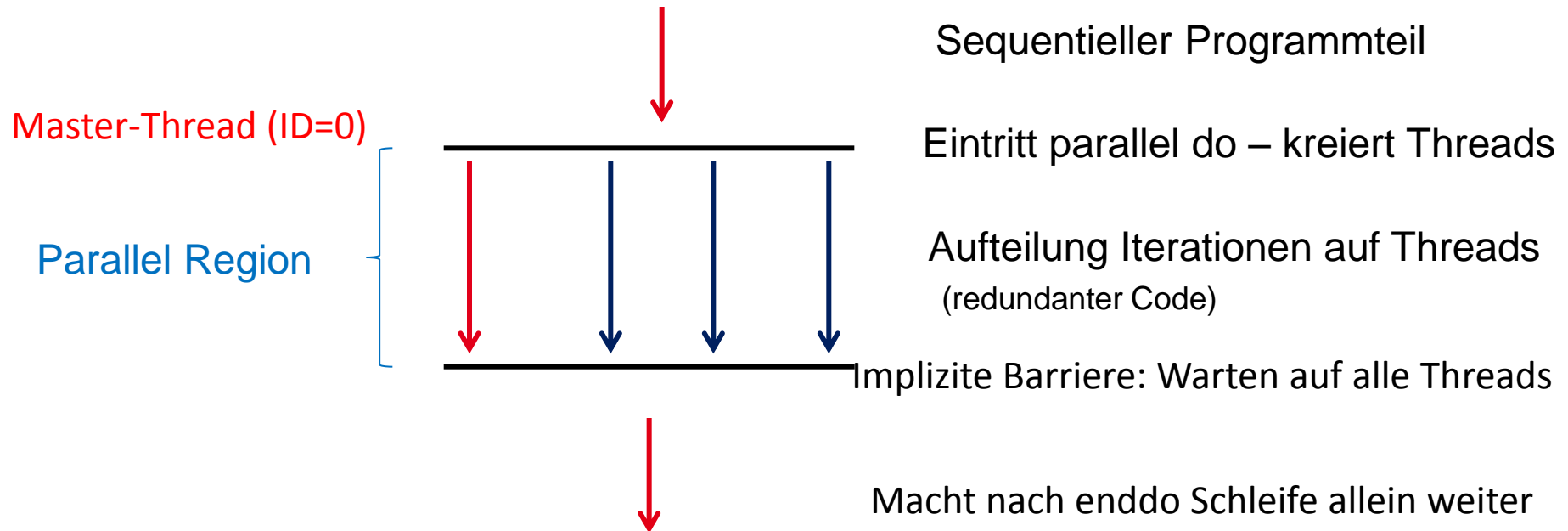
return

end



OpenMP Fork-Join Programmier Model

OpenMP Merkmale!





OpenMP – Allgemeine Einführung IV

Eine Stärke von OpenMP ist der inkrementelle Einsatz in sequentiellen Programmen:

OpenMP Direktiven werden in einem speziellen Format angegeben,
die nur der OpenMP Compiler versteht,
für den regulären FORTRAN Compiler aber nur als Kommentare interpretiert werden.

Erster Schritt für OpenMP Implementierung:

Finde Bereiche zur Parallelisierung im sequentiellen FORTRAN Programm !

Dann **inkrementelle Einarbeitung** von OpenMP in existierendes FORTRAN Programm.

(Vorteil: rein sequentielles Debugging immer noch möglich)



OpenMP Aufruf und Notation

Aufruf eines OpenMP Programmes:

```
gfortran -fopenmp -o hello hello.f90
```

OpenMP Notation im FORTRAN Code:

```
!$OMP Direktive [Bestimmung1], [Bestimmung2], ..., [Bestimmung n]
```



OpenMP - Einführungsbeispiel I

Subroutine mxv (m,n,a,b,c) ! Matrix-Vektor Produkt

implicit none

integer (kind=4) :: m, n

real (kind = 8) :: a(1:m), b(1:m,1:n), c(1:n)

integer :: i,j

!\$OMP PARALLEL DO DEFAULT(NONE) &

!\$OMP SHARED (m,n,a,b,c) PRIVATE(i,j)

 do i = 1,m

 a(i) = 0.0

 do j = 1,n

 a(i) = a(i) + b(i,j) * c(j)

 enddo

 enddo

!\$ OMP END PARALLEL DO

....



OpenMP Einführungsbeispiel II

Subroutine mxv (m,n,a,b,c)

! Matrix-Vektor Produkt

.....

!\$OMP PARALLEL DO DEFAULT(NONE) &

!\$OMP SHARED (m,n,a,b,c) PRIVATE(i,j)

do i = 1,m

 a(i) = 0.0

 do j = 1,n

 a(i) = a(i) + b(i,j) * c(j)

 enddo

enddo

!\$OMP END PARALLEL DO

....

} parallele Region

OpenMP kann angewendet werden da:

* Iterationen voneinander unabhängig sind

z.B. $vv(i,j) = vv(i+1,j) + a(i,j)$ **geht nicht!**

hier würde der Wert der i+1 Iteration genutzt

d.h. **die Datenabhängigkeiten sind zu prüfen!**

* die OpenMP Direktiven beziehen sich auf

„strukturierte Blöcke“ von FORTRAN Code

d.h. es darf keine Verzweigung (Branch)

in oder aus der parallelen Region geben



OpenMP Einführungsbeispiel III

Subroutine mxv (m,n,a,b,c)

.....

```
!$OMP PARALLEL DO DEFAULT(NONE) &
!$OMP SHARED (m,n,a,b,c) PRIVATE(i,j)
```

```
do i = 1,m
```

```
    a(i) = 0.0
```

```
    do j = 1,n
```

```
        a(i) = a(i) + b(i,j) * c(j)
```

```
    enddo
```

```
enddo
```

```
!$OMP END PARALLEL DO
```

....

Die OpenMP Direktive (PARALLEL DO) bezieht sich auf die direkt nachfolgende FORTRAN Zeile (do i =1,m)

und parallelisiert damit auch nur die i-Schleife.

Verschachtelungen sind möglich.

Wichtig ist die Definition der Nachfolgezeile mittels & in FORTRAN Notation!

(Syntaxfehler werden ohne Warnung ignoriert!)



OpenMP Einführungsbeispiel IV

Subroutine mxv (m,n,a,b,c)

.....

!\$OMP PARALLEL DO DEFAULT(NONE) &

!\$OMP SHARED (m,n,a,b,c) PRIVATE(i,j)

do i = 1,m

a(i) = 0.0

do j = 1,n

a(i) = a(i) + b(i,j) * c(j)

enddo

enddo

!\$OMP END PARALLEL DO

....

Die OpenMP Direktive beinhaltet ebenfalls die Regelung der Zugriffsrechte:

DEFAULT (NONE) überlässt dem User die persönliche Festlegung der Zugriffsrechte

SHARED (m,n,a,b,c) erlaubt allen Threads den gleichzeitigen Zugriff auf die gelisteten Variablen

PRIVATE(i,j) setzt die beiden Variablen private, so dass jeder Thread nur Zugang zu einer lokalen, einmaligen Kopie der Variablen hat.



OpenMP Thread Abfrage als Funktionsaufruf

Folgende OpenMP Funktionen ermöglichen Informationen zu den Threads:

(Im Programm ist dann die Bibliothek mit `use omp_lib` einzubinden)

`OMP_GET_NUM_THREADS ()` Anzahl der arbeitenden (in use) Threads

`OMP_GET_THREAD_NUM()` ID des aktuellen Threads

`OMP_GET_MAX_THREADS()` Maximale Anzahl der verfügbaren Threads



OpenMP Bestimmung der Thread Anzahl I

Folgende OpenMP Syntax wird für die Auswahl der Threadanzahl verwendet:

A) im Aufruf des Programmes z.B. im Makefile:

```
gfortran -fopenmp -o hello hello.f90
```

Kompilieren

```
export OMP_NUM_THREADS=4
```

Umgebungsvariable setzen (unter bash)

bzw.

```
setenv OMP_NUM_THREADS 4
```

Umgebungsvariable setzen (unter csh)

```
./hello
```

Ausführung



OpenMP Bestimmung der Thread Anzahl II

Folgende OpenMP Syntax wird für die Auswahl der Threadanzahl verwendet:

B) Aufruf über Subroutine Call im Programm selbst z.B. :

Call OMP_SET_NUM_THREADS (4)

Dazu muss vorher die OpenMP Anbindung im Makefile erfolgen:

`gfortran -fopenmp -o hello hello.f90` Kompilieren



OpenMP Übersicht

- Stärke von OpenMP ist der inkrementelle Einsatz in sequentiellen Programmen
(nach Fork-Join Programmier-Modell)
- Erster Schritt in Anwendung: Definition von Bereichen zur Parallelisierung
- Dann prüfen: Datenabhängigkeit
Regelung der Zugriffsrechte
- Hinweis zur Syntax: Definition Nachfolgereihe in FORTRAN Notation



Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG



Danke das wars!