8. HPF – High Performance FORTRAN

- datenparallele Erweiterung von FORTRAN 90
- SPMD-Ansatz
- daten-parallele Array-Operationen
- verteilte Datenstrukturen
- Schleifenparallelität

FORTRAN 77 -> FORTRAN 90 -> HPF

- FORTRAN 90 = FORTRAN 77 +
  - Zeiger (pointers)
  - Rekursion
  - benutzerdefinierte Datenstrukturen
  - ...
  - Matrizenoperationen , z.B. A = B + C, A = B * C, A = B – C
- HPF = FORTRAN 90 +
  - Direktiven (= strukturierte Kommentare zur Compilersteuerung)
    - Processors pr(20) pr(5;4)
    - Align
    - Distribute

Datenparallelität in FORTRAN 90

- FORTRAN 90 hat keine explizite Parallelität, aber implizite Datenparallelität über Arrays, meist Vektoren (eindimensional) und Matrizen (zweidimensional).

- Terminologie für Arrays:
  - Rang: Anzahl von Dimensionen
  - Vektor: eindimensionales Array
  - Form (shape):
    Vektor, der die Anzahl der Elemente für jede Dimension angibt
    Z.B.:

```
INTEGER I (10)                    ! Rang 1, Form (10)
REAL, DIMENSION (8,8) :: A, B     ! Rang 2, Form (8,8)
REAL C (0:7, 0:7)                 ! Rang 2, Form (8,8)
```

    A, B und C passen zueinander, wenn sie dieselbe Form haben.

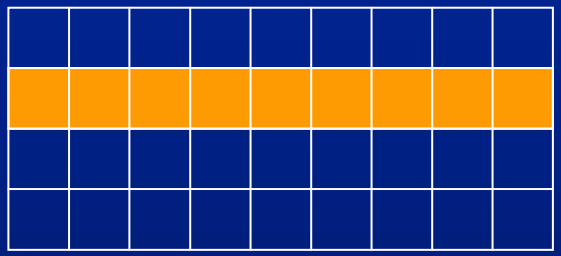
Operationen können auf passende Arrays angewendet werden.

Datenparallele Arrayoperationen

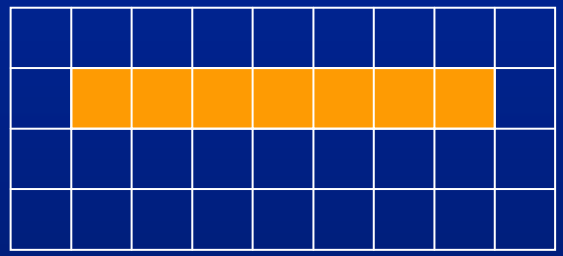
- binäre Operationen
  - B+C, B*C, B-C
- unäre Operationen
  - A = sqrt(B)
- Array-Skalar-Operationen
  - B+c
  - B*c
- Reduktionsoperationen
  - SUM(A)
  - MAXVAL(A)
  - MAXLOC(A)
- Restrukturierungsoperationen
  - TRANSPOSE(A)
  - CSHIFT(A) (circular shift)

Sektionsnotation für Arrays

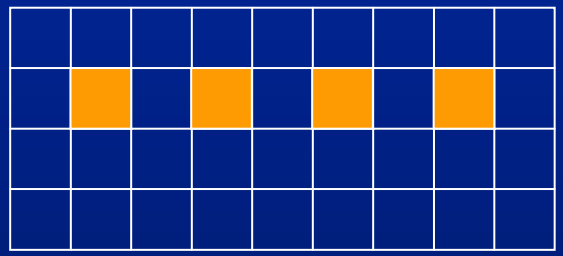
array (...., ini : end : offset/step , ...)



A (2, :)



A (2, 2:8)



A (2, 2:9:2)

- **Defaultwerte sind:**
  ini = untere Indexgrenze, end = obere Indexgrenze, offset = 1
  A(:) ist also gleichbedeutend zu A.

Irreguläre Arraysektionen

- Irreguläre Arraysektionen werden durch Vektoren spezifiziert:

  Beispiel: Falls V = (/5, 2, 1, 2/) bezeichnet
  A(V) die Wertfolge (/A(5), A(2), A(1), A(2)/).

- Arraybezeichnungen können verschiedene Sektionsnotationen enthalten, z.B. B(3:5:2,2,V).

- Rang und Form von Arrays werden durch die Dimensionen mit Sektionsindizes bestimmt – skalare Indizes spielen keine Rolle.

  Beispiel: B(3:5:2, 2, V) mit V = (/8, 1/) hat
  den Rang 2 und die Form (2,2).

  Die Elemente sind: [B(3,2,8) B(3,2,1)]
  [B(5,2,8) B(5,2,1)]

Datenparallele Arrayzuweisungen

In var = expr
wird expr vollständig für alle Indizes vor der Zuweisung ausgewertet, d.h. in expr werden „alte“ Werte verwendet.

Beispiel:

A(2:3) = A(1:2) bewirkt A = (/ „altes“ A(1), „altes“ A(1), „altes“ A(2), ... /)

Zum Vergleich:

```
DO i=2,3
    A(i) = A(i-1)
ENDDO
```

bewirkt A = (/ „altes“ A(1), „altes“ A(1), „altes“ A(1), ... /)

Letzteres ist gleichbedeutend mit A(2:3) = A(1).

Weiteres Beispiel: A(2:9) = 0.5 * (A(1:8) + A(3:9)).

Logische Array-Ausdrücke (Masken)

- Vergleichsoperatoren: > >= < <= == /=
- Logische Operatoren: .AND. .OR. .EQV. .NEQV.

- Beispiel: Falls V = (/ 1, 0, -5 /)
  ergibt (V > 0) den Vektor (/ .TRUE., .FALSE., .FALSE. /).

- Maskierung von Array-Zuweisungen mittels WHERE-Konstrukt:

  Beispiele:

```
REAL A(10), B(10)
WHERE (A>0.0) B = 1.0/A
```

```
WHERE (A > 0.0)
    B = 1.0 / A
ELSEWHERE
    B = 0.0
ENDWHERE
```

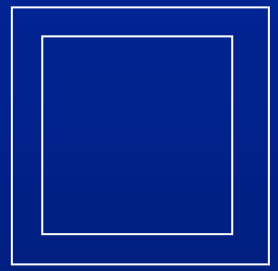
Array Funktionen

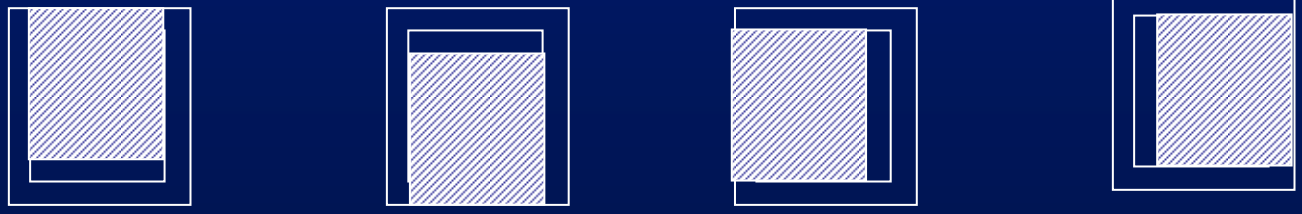
Benutzerdefinierte Funktionen können Array-Resultate berechnen:

```
FUNCTION matvec (M, V)
      REAL M(:,:), V(SIZE(M,2)), matvec (SIZE(M,1))
      INTEGER i
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - !
!  Returns matvec(i) = SUM_j {M(i,j) * V(j)}                                    !
! - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - !
      DO  i = 1, SIZE (matvec)
            matvec(i) = SUM (M(i,:) * V(:))
      ENDDO
END FUNCTION
.....
REAL A(10,10), V(10)    ! Initialisiere A und V!

V = matvec (A,V)
```

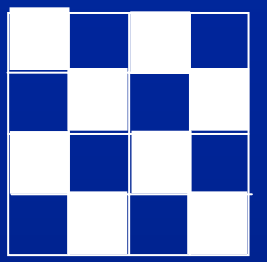
Beispiel: Jacobi Iteration

```
REAL, DIMENSION (m,n) :: a, old_a

old_a = 0.0
CALL init(a)          ! Rand von a festlegen
a (2:m-1, 2:n-1) = 0.0    ! Inneren Bereich auf 0.0 setzen

DO WHILE (ANY (a-old_a > 1E-07 * a))
    old_a = a
    a (2:m-1, 2:n-1) = 0.25 *                    &
        (a(1:m-2,2:n-1) + a(3:m, 2:n-1) + a(2:m-1,1:n-2) + a(2:m-1, 3:n))
ENDDO
```





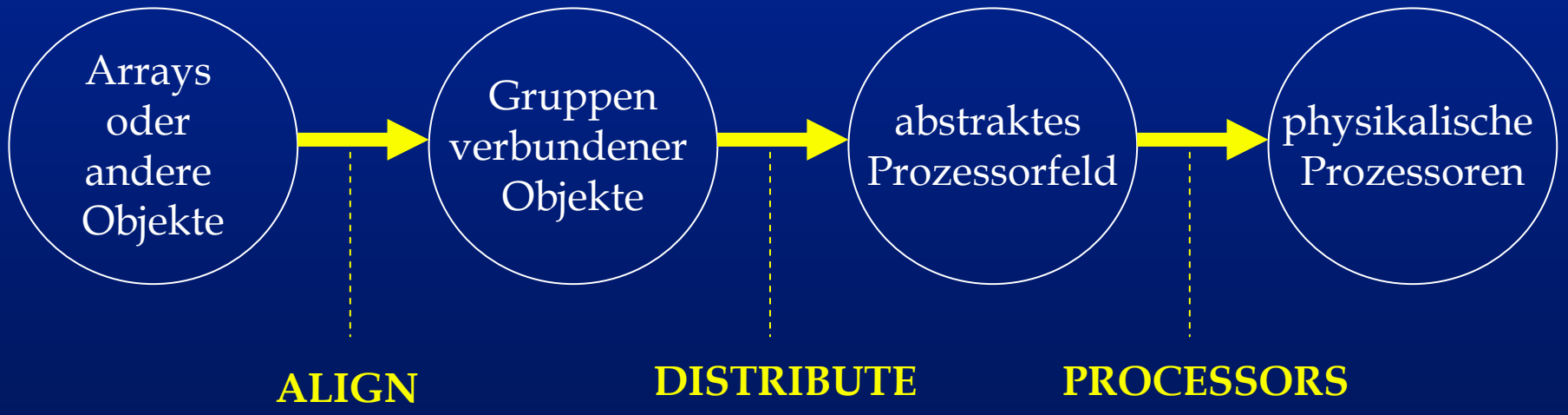
Beispiel: Rot-Schwarz-Relaxation



```
REAL, DIMENSION (m,n) :: a, old_a
LOGICAL :: even (m-2, n-2)
old_a = 0.0
CALL init(a, even)          ! Rand von a festlegen
   ! Initialisiere even(i,j) mit .TRUE., falls i+j gerade, .FALSE. sonst

DO WHILE (ANY (a-old_a > 1E-07 * a))
   old_a = a
   WHERE (even)
      a (2:m-1, 2:n-1) = 0.25 *                    &
         (a(1:m-2,2:n-1) + a(3:m, 2:n-1) + a(2:m-1,1:n-2) + a(2:m-1, 3:n))
   ELSEWHERE
      a (2:m-1, 2:n-1) = 0.25 *                    &
         (a(1:m-2,2:n-1) + a(3:m, 2:n-1) + a(2:m-1,1:n-2) + a(2:m-1, 3:n))

   ENDWHERE
ENDDO
```

Ausführungsmodell von HPF

- **Die HPF-Direktiven steuern die Abbildung von Daten auf Prozessorspeicher**
- **Zweistufige Abbildung:**



HPF Datenverteilungsdirektiven

- !HPF$ PROCESSORS proc_name(dim1,dim2,..., dimN) deklariert ein abstraktes Prozessorfeld.
  - proc_name ist der Name des Prozessorfelds
  - dim1 ... dimN bestimmen die Größe und Form des Prozessorfelds
- !HPF$ ALIGN array WITH target verknüpft zwei Felder
- !HPF$ DISTRIBUTE list_of_arrays [ONTO proc_name] verteilt Felder auf ein Prozessorfeld

Die ALIGN-Direktive

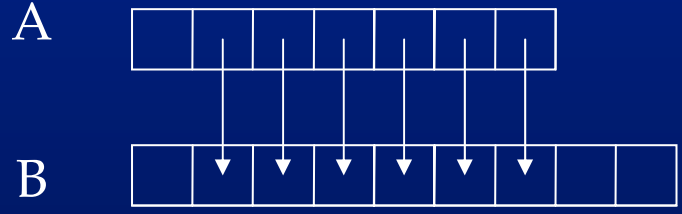
- **ALIGN array WITH array**

reference

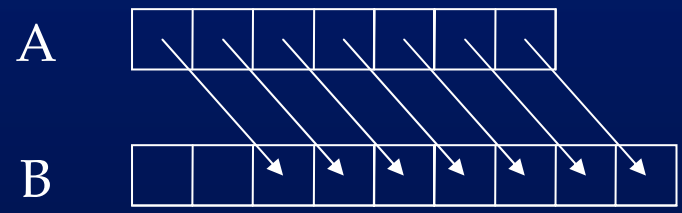
bewirkt die Verteilung von Arrays in Bezug auf ein Referenzarray.

Beispiele:

- **ALIGN A(I) WITH B(I)**

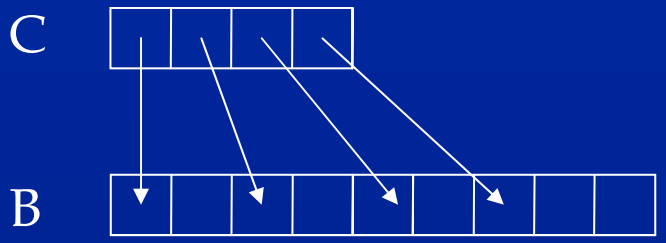


- **ALIGN A(I) WITH B(I+2)**

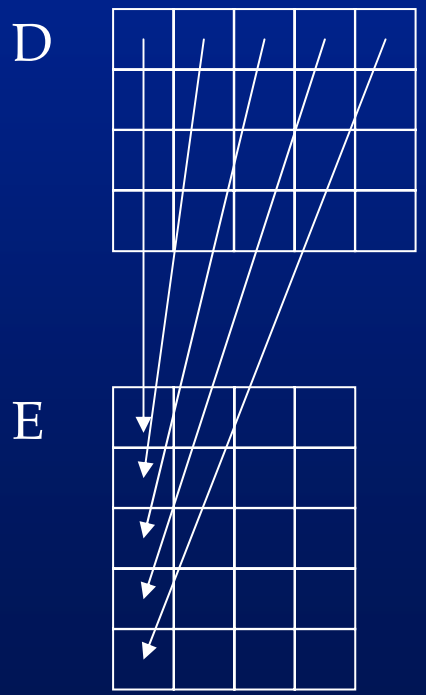


Weitere Beispiele

- ALIGN C(I) WITH B(2*I)

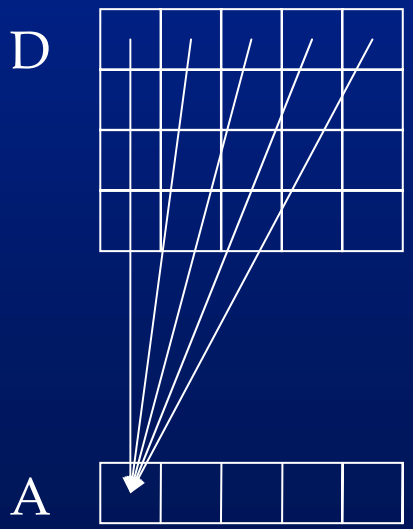


- ALIGN D(I,J) WITH E(J,I)

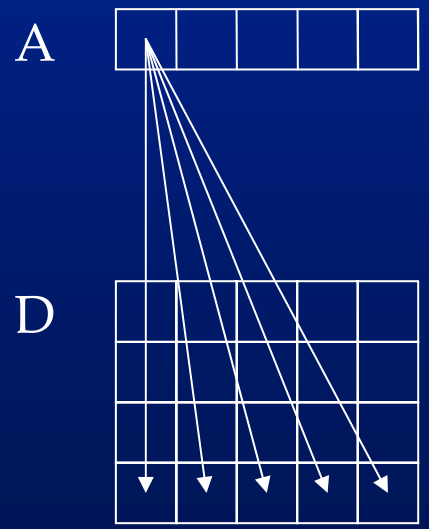


Transposition

- ALIGN D(:,*) WITH F(:)



- ALIGN F(:) WITH D(*,:)

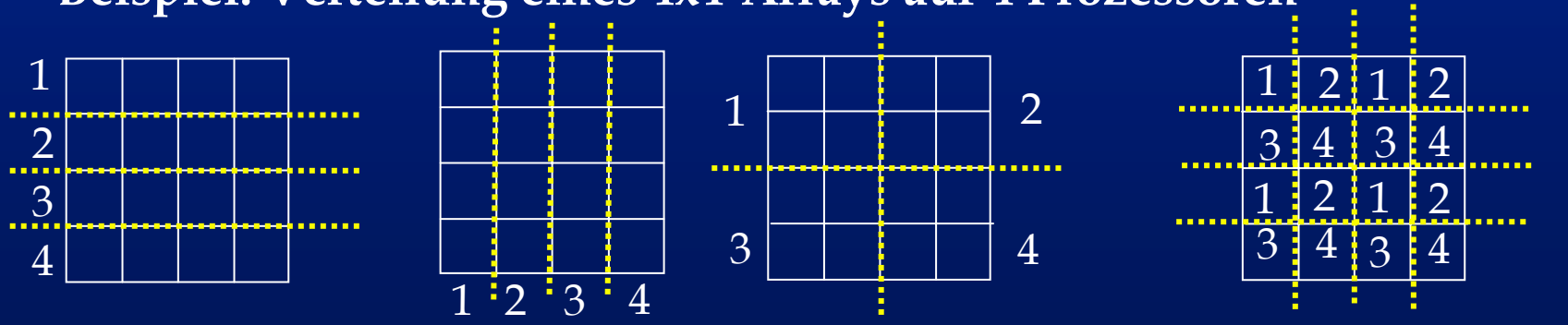


Replikation

Die Distribute-Direktive

- **DISTRIBUTE arrays [ONTO proc_name]**
  dient der Verteilung von Arrays auf ein abstraktes Prozessorfeld
  - **blockweise => zusammenhängende, etwa gleich große Blöcke**
  - **zyklisch => reihum Verteilung**
  - *** => keine Verteilung in entsprechender Dimension**

Beispiel: Verteilung eines 4x4-Arrays auf 4 Prozessoren



eindimensionales Prozessorfeld

zweidimensionales Prozessorfeld

Beispiel: Jacobi mit HPF Direktiven

```
      REAL, DIMENSION (m,n) :: a, old_a
!HPF$ PROCESSORS pr(4)
!HPF$ ALIGN a (:,:) WITH old_a(:,:)
!HPF$ DISTRIBUTE old_a(BLOCK,*) ONTO pr

      old_a = 0.0
      CALL init(a)              ! Rand von a festlegen
      a (2:m-1, 2:n-1) = 0.0    ! Inneren Bereich auf 0.0 setzen

      DO WHILE (ANY (a-old_a > 1E-07 * a))
        old_a = a
        a (2:m-1, 2:n-1) = 0.25 *                    &
            (a(1:m-2,2:n-1) + a(3:m, 2:n-1) + a(2:m-1,1:n-2) + a(2:m-1, 3:n))
      ENDDO
```

Weiteres Beispiel: Pairwise Interactions

```
      program hpf_pairwise_interactions
!HPF$ PROCESSORS pr(10)
      real X(3,1000), Force(3,1000), Tmp(3,1000)
!HPF$ ALIGN Force(:,:) WITH X(:,:)
!HPF$ ALIGN Tmp(:,:) WITH X(:,:)
!HPF$ DISTRIBUTE X(*,BLOCK) ONTO pr
      Force = 0.0
      Tmp = X
      do i = 1, 999
            tmp = CSHIFT(Tmp,1,2)
            Force = Force + f(X,Tmp)
      enddo
      end
```

Schleifenparallelität mit FORALL

FORALL (range_1, ..., range_n, boolean_exp) assignment

Beispiele:

- FORALL (i=2:9) A(i) = 0.5*(A(i-1) + A(i+1))
  ist gleichbedeutend mit A(2:9) = 0.5 * (A(1:8) + A(3:10))
- FORALL (i=1:10, A(i)>0.0) A(i) = 1/A(i)
  ist gleichbedeutend mit WHERE (A(1:10)>0.0) A(1:10) = 1.0/A(1:10)
- FORALL (i=1:m, j=1:n) even(i,j) = (MOD (i+j, 2) == 0)
- FORALL (i=1,n) A(i,i) = ... ! Diagonale
  FORALL (i=1,n, j=1,m, j>= i) = ... ! oberes Dreieck
- FORALL (i=1,n, j=1:m) &
  C(i,j) = DOT_PRODUCT (A(i,:), B(:,j))

Die Independent-Direktive

Die INDEPENDENT-Direktive zeigt an, dass die Iterationen einer Do-Schleife unabhängig voneinander sind und daher parallel ausgeführt werden können.

Beispiel:

```
!HPF$   INDEPENDENT
        DO i = 1, n1
!HPF$   INDEPENDENT
                DO j=1,n2
                        DO k=1,n3
                                A(i,j) = A(i,j) + B(i,j,k)
                        ENDDO
                ENDDO
        ENDDO
```