Présentation du langage HPF
High Performance Fortran

Fabien COELHO

—

Centre de Recherche en Informatique

École des mines de Paris

—

Renpar'7, Mons, 30 mai 1995

CRI, ENSMP

Centre de Recherche en Informatique

École des mines de Paris

Compilation pour machines parallèles

- parallélisation automatique (projet PIPS)
- analyse statique de programme

  (complexité, effets, dépendances...)
- génération de transferts de données
- compilation de HPF

PIPS

Paralléliseur Interprocédural de Programmes Scientifiques

- parallélisateur expérimental
- source à source Fortran
- financé par la DRET (DGA) depuis 1988
- utilisation d'informations interprocédurales
- gros programme modulaire (100 000 lignes de C)

$\boxed{\text{mémoire partagée} \rightarrow \text{mémoire répartie}}$

Plan

Enjeux

HPF

- historique

- placement des données

- parallélisme

- reste

Compilation

- problèmes

- prototypes

Conclusion

le calcul scientifique aujourd’hui

- baisse des coûts des supercalculateurs
- augmentation des puissances maximales proposées
- adaptation des métiers à ces outils

$\boxed{\Rightarrow \textbf{expansion}}$

du club de golf aux prévisions météorologiques. . .

Teraflop/s?

conditions d'accès aux **Grand Challenges**

- objectifs du High Performance Computing and Communitions Program du National Science Fundation (NSF)
- programme de recherche US pour le développement technologique US

mais

marché ouvert pour stations performantes (RISC 6000)

Grand Challenges

– climatologie, circulations océaniques, pollution

– turbulences, systèmes de combustion

– génome humain

– chromodynamique quantique

– étude des semiconducteurs et des supraconducteurs

– vision et cognition

not so Grand Challenges

utilisation industrielle des supercalculateurs :

– analyse structurelle d'un embout de club de golf en Titane.

– simulation de l'écoulement dans une cuvette de toilettes pour diminuer consommation d'eau et bruit.

– étude de l'écoulement de savon liquide pour le remplissage de bouteilles.

Évolution des matériels

séquentiel

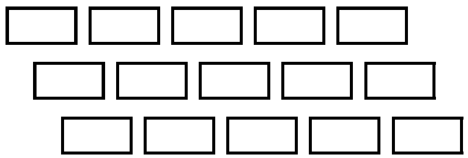
les instructions se suivent. . .

de plus en plus vite. . .

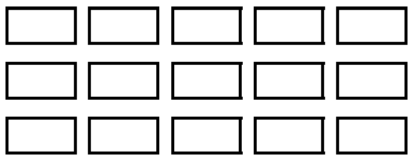
mais les limites approchent. . .

→ parallélisme

Parallélisme monoprocesseur



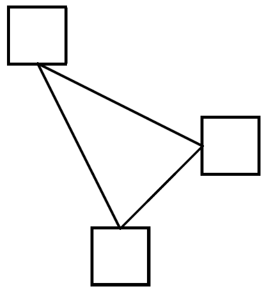
pipeline



superscalaire, VLIW

Parallélisme multiprocesseur

réseau de
processeurs



Architectures multiprocesseurs

contrôle :

- SIMD : flot unique (CM2, *Thinking Machine*)
- MIMD : flots multiples (CS, *Meiko*)

mémoire

- SM : partagée (Sparc Server 1000, *Sun*)
- DM : répartie (CM5, *Thinking Machine*)

MIMD-DM

principe **extensible** *(scalable)*

programmation par échange de messages

- programmation complexe
- productivité faible
- portabilité réduite

$\boxed{\Rightarrow \text{coût important}}$

Solutions

- **SVM** *(Shared Virtual Memory)* : *hard, OS*
  - KSR-1, *Kendall Square Research*
  - KOAN, *IRISA, Intel iPSC/2*
- **langages** parallèles : compilateur, *soft*
  - Vienna Fortran, Fortran D, CM-Fortran...
  - Pandore, Kali, *Id Nouveau*, *Crystal*...

portabilité : directives de compilation, standards

l'héritage Fortran

– dusty deck

– habitudes des scientifiques

– standards :

Fortran 77

Fortran 90 (structuré, data-parallèle)

Parallel Computing Fortran (mémoire partagée)

cependant il existe d'autres approches : **Sisal**

Nouvelles Extensions

- basées sur Fortran
- parallélisme de données
- directives de compilation

Standardisation de facto

- portabilité pour les utilisateurs
- argument de vente pour les constructeurs

$\boxed{\text{HPF !}}$

High Performance Fortran Forum

– initiative de DEC fin 91

– constructeurs, universitaires, utilisateurs

– pour définir un standard *de facto*

– réunions pleinières sur 92-début 93

– reprise des discussions en janvier 94

$\boxed{\rightarrow \textit{High Performance Fortran}}$

Qui ?

- **Construteurs** : IBM, TMC, Cray, Fujitsu, DEC, Archipel, Convex, Maspar, Meiko, nCube, Sun, HP...
- **Universités** : Rice, Yale, Cornell, Stuttgart, Vienna...
- **Laboratoires** : LANL, LLNL, ICASE, RIACS...
- **Utilisateurs** : Shell, Schlumberger...

Europe

- Hans Zima, Barbara Chapman (Vienna U)
- John Merlin (Southampton U)
- Clemens-August Thole (GMD)
- James Cownie (Meiko), Henk Sips (Delft)...

France

- Jean-Laurent Philippe (Archipel)
- Jean-Eric Waroquier (ACSET, filiale ACRI)

Objectifs du Forum

– créer un nouveau standard Fortran,

– performant pour les Machines Massivement Parallèles,

– MIMD ou SIMD à mémoire non uniforme (NUMA)

– adaptable à de nombreuses architectures.

Buts du Forum

- proposition finale novembre 92
- proche des autres standards (F77, F90)
- simplicité
- interfaçage et ouvertures
- critères de validation
- compilateurs disponibles à court terme

Organisation

- réunions régulières aux US
- travail en sous groupe

  (data distribution, forall, f90, instrinsics, I/O, ...)
- courrier électronique permanent
- votes à la majorité des organismes assidus !

Base de départ

- **Standards** : Fortran 77, Fortran 90, PC-Fortran.
- **Constructeurs** : CM-Fortran, MPP-Fortran, ...
- **Recherche** : Fortran D, Vienna Fortran.

Idées retenues

à partir des exemples existants :

- **directives** de compilation en commentaires
  (pas de modifications sémantiques. . .)
- programme séquentiel + placement des données
- sections de tableau (F90)
- concepts d'**alignement** et **distribution**
- « sémantiques » de boucles parallèles

Vie du Forum

- grandes batailles (Templates or not)
- projets abandonnés (I/O)
- principes oubliés (simples directives dépassées)
- quelques retards et transformations de dernière minute

High Performance Fortran

spécification finale du 3 mai 1993

- un Fortran pour calculateurs parallèles
- parallélisme de données
- basé sur Fortran 90
- directives de placement de données
- constructions parallèles
- fonctions intrinsèques (réductions. . .)
- sous-ensemble officiel

Modèle HPF

efficace et portable

deux facteurs essentiels interviennent pour produire des codes performants :

- le **parallélisme** des calculs à exploiter
- les **communications** qu'ils nécessitent

HPF permet d'ajuster les programmes :

- avec des constructions parallèles
- des directives de placement de données

Directives

les directives, déclaratives ou exécutables, sont placées en commentaires Fortran :

- `!HPF$` *directive*
- `CHPF$` *directive*
- `*HPF$` *directive*

de plus :

- directives sur plusieurs lignes (&)
- les blancs y sont significatifs

Modèle de placement de données

Groupe de Guy L. Steele Jr., TMC.

Objectifs

- spécifier le placement des données sur les différents processeurs d'une machine à mémoire non uniforme.
- **utilisateur** : compréhensible, intuitif
- **compilateur** : optimisable
- **applications** : pertinence des placements proposés

Survol

- modèle HPF
- directives de placement (`ALIGN`, `DISTRIBUTE`)
- quelques remarques
- interfaces pour fonctions et procédures
- conclusion sur le placement

Modèle

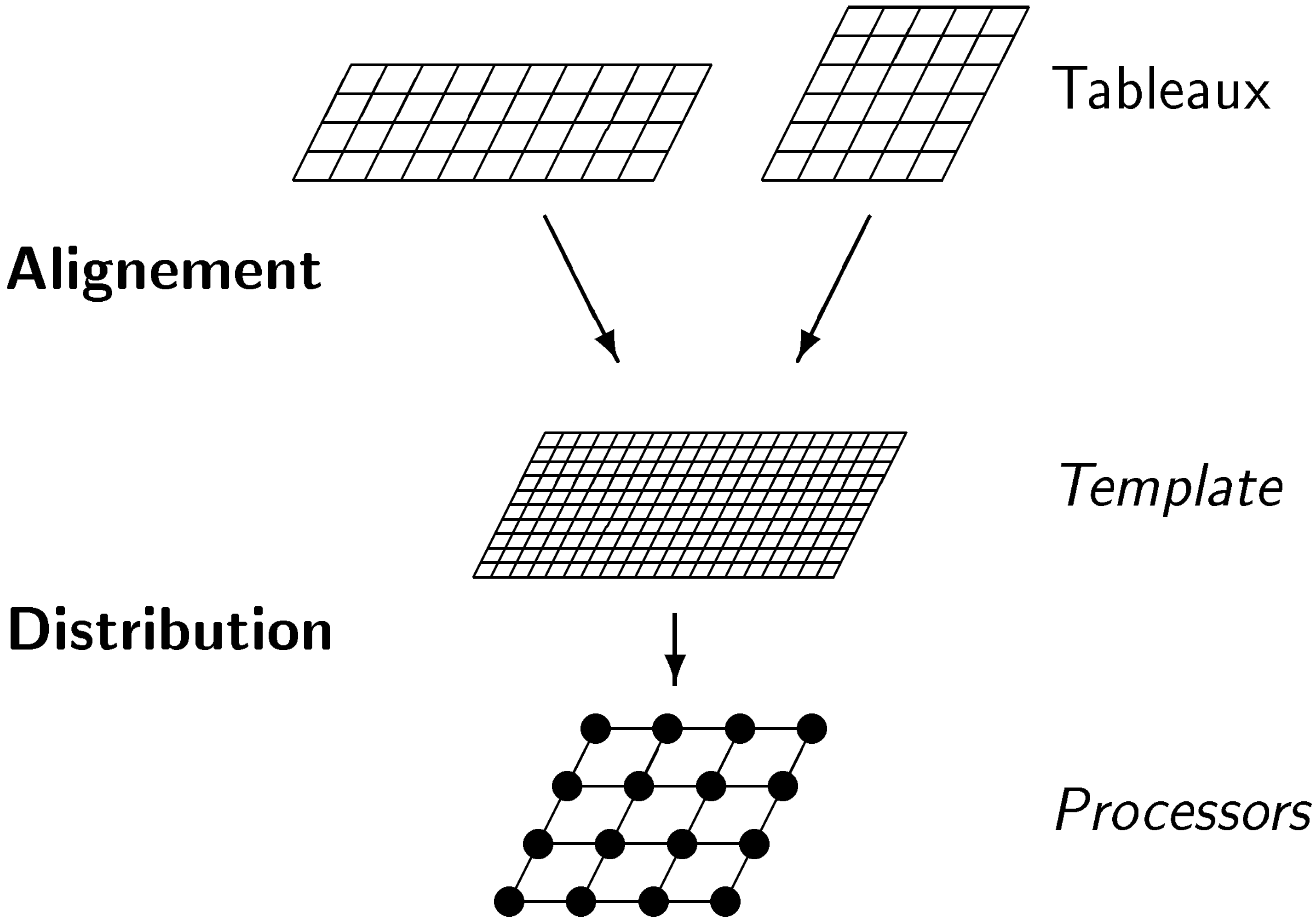
3 niveaux entre les tableaux de données et les processeurs physiques :

- les *tableaux* sont **alignés** sur des *templates*
- les *templates* sont **distribués** sur des *processors*
- les *processors* sont placés sur les processeurs physiques

seuls les **2** premiers niveaux sont spécifiés

directives : `ALIGN`, `TEMPLATE`, `DISTRIBUTE`, `PROCESSORS`

Modèle de placement HPF



Placement à 2 niveaux

équilibre calculs, communications minimales

Alignement affine (`ALIGN`)

- mise en correspondance des données *(réplication)*
- lié au problème *(localité des données)*

Distribution régulière (`DISTRIBUTE`)

- découpage pour les processeurs `block cyclic(n)`
- lié à l'architecture

  (granularité du parallélisme, équilibrage de charge)

Templates

`TEMPLATE` est une directive de déclaration

- les *templates* sont des tableaux de *rien*
- alignement sur un espace plus grand que tous les tableaux du problème
- tableaux utilisés comme templates implicites

```
CHPF$ TEMPLATE T1(100,100), T2(100)
```

Template or not Template

Un long débat a animé le groupe `hpf-distribute` sur l'adoption des templates. Il opposait :

– Hans Zima

(Vienna Fortran, qui ne comporte pas de *templates*)

– Rice University (Ken Kennedy)

(Fortran D, donc les *decompositions* sont très proches des *templates*)

L'Amérique a gagnée sur son terrain !

Pour

– plus grand qu'un tableau

– portage plus facile entre machines

– tableaux mieux liés ensemble

Contre

– n'ajoute rien à l'expressivité

– objet de plus à gérer

– désoriente l'utilisateur

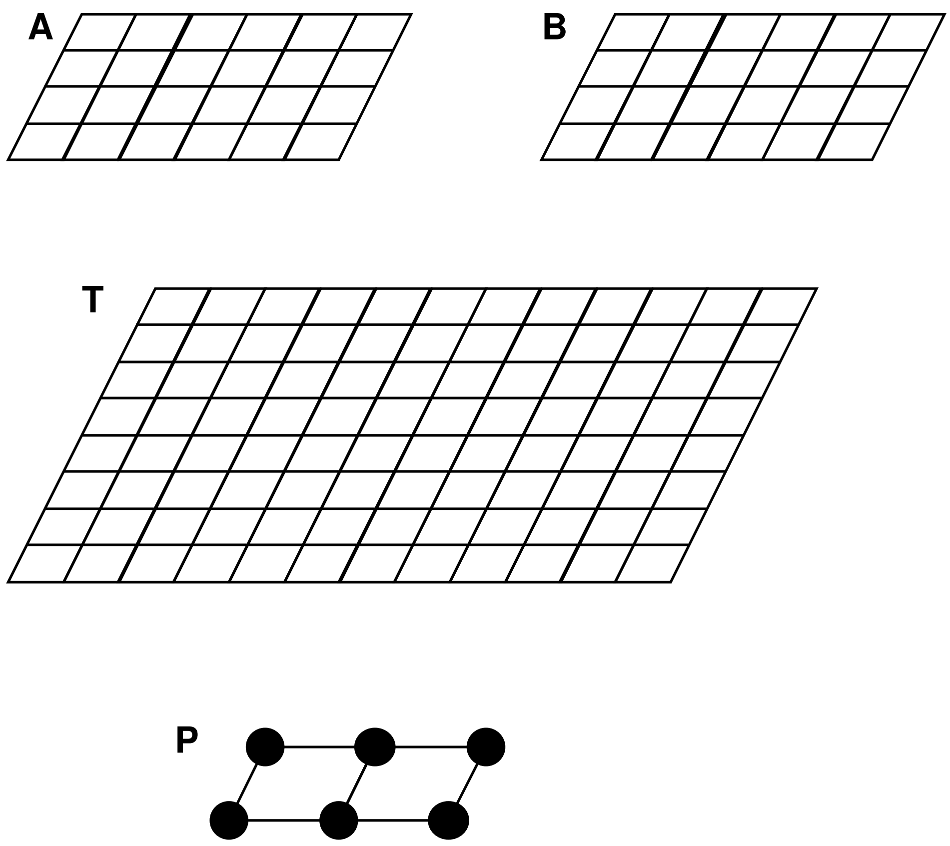
Alignement

`ALIGN` est une directive statique

- les *tableaux* sont **alignés** sur des *templates* :
- 2 éléments de tableau alignés sur un même élément de template sont supposés se retrouver finalement sur le même processeur.

```
CHPF$ ALIGN A(I...) WITH T(affine(I)...)
```

Objets HPF




```
array A(6,4), B(6,4)
template T(12,8)
processors P(3,2)
```

Contraintes

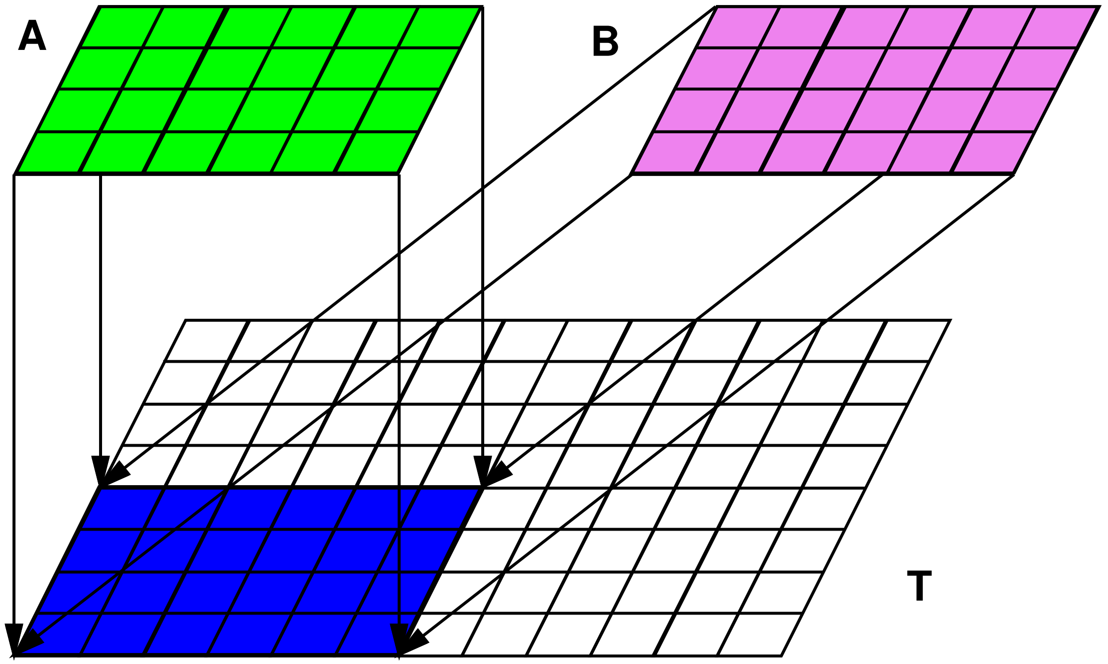
- fonction affine
  - d'une seule variable muette (`a*I+b`)
  - entière (`a` et `b` entiers)
- une seule utilisation d'une variable muette :

  pas de `ALIGN A(I) WITH T(I,I)`...

Possibilités de l'Alignement

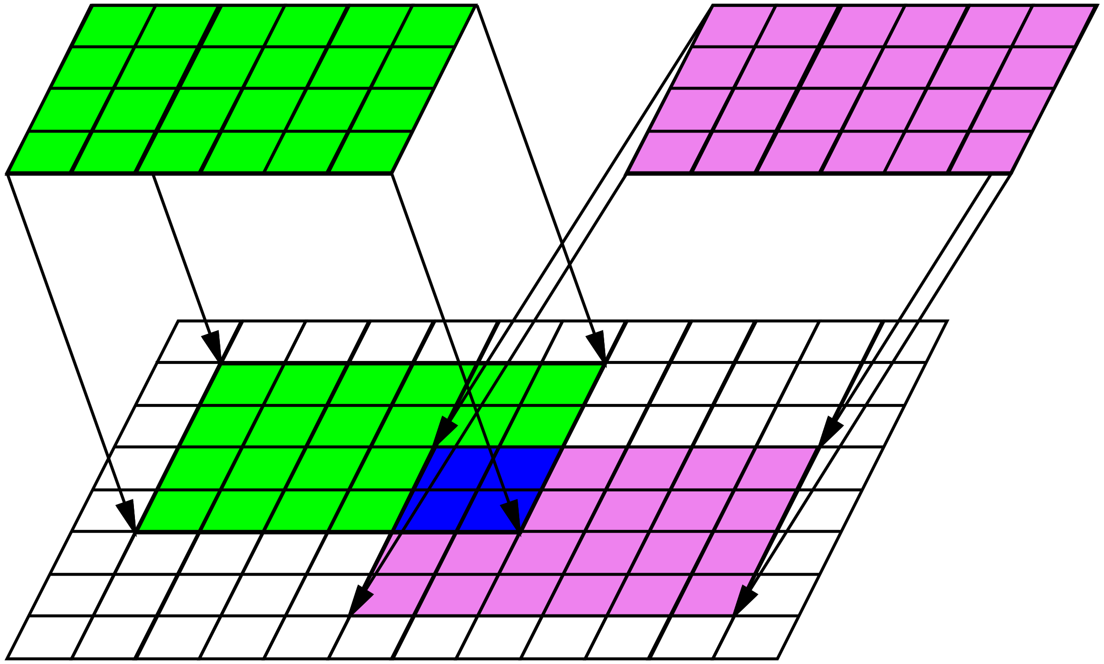
- correspondance **directe**
- **transposition** d'un tableau par rapport à l'autre
- **duplication** sur une dimension du template
- **écrasement** d'une dimension de tableau
- **éclatement** régulier du tableau
- ...

Identité



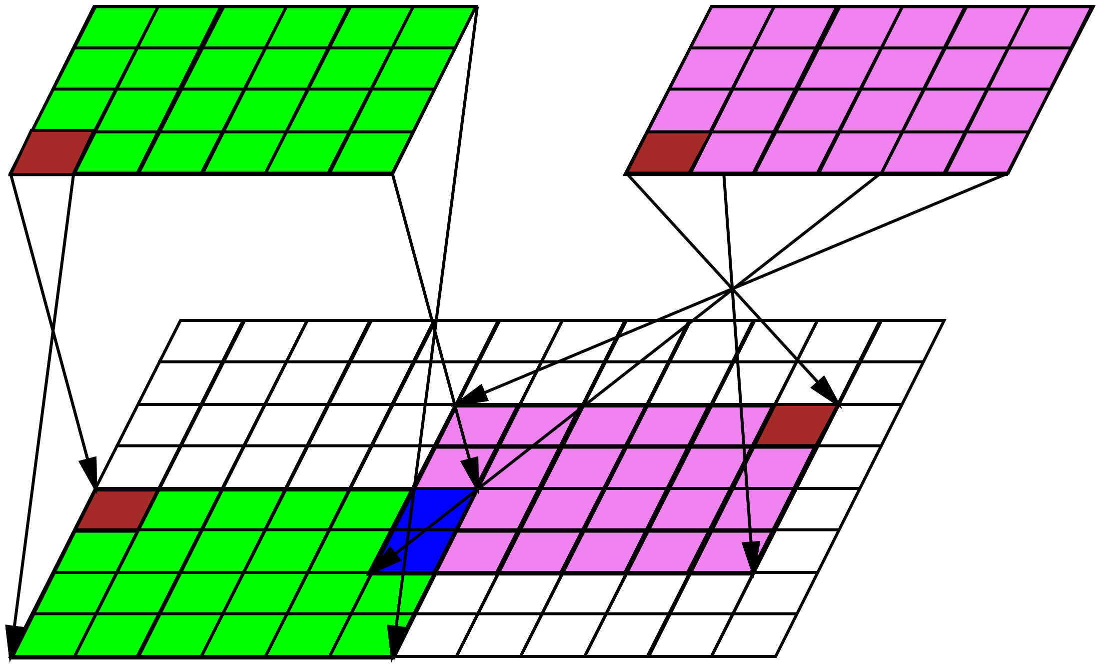

```
align A(i,j) with T(i,j)
align B with T
```

Décalage



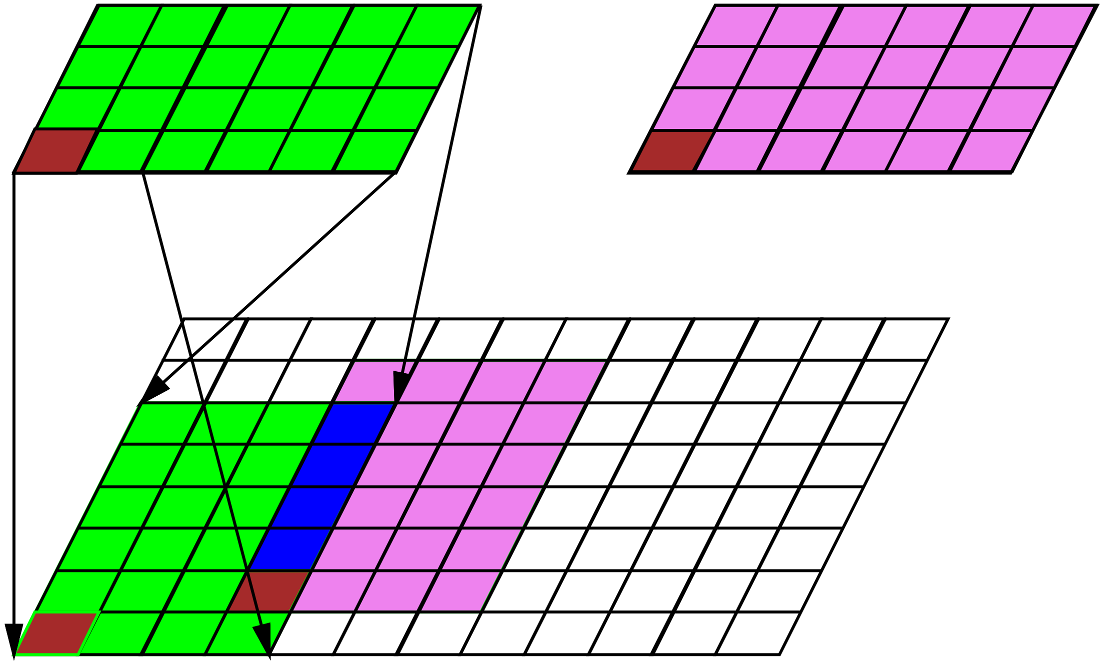
```
align A(i,j) with T(i+1,j+3)
align B(:,:) with T(6:11,2:5)
```

Retournements



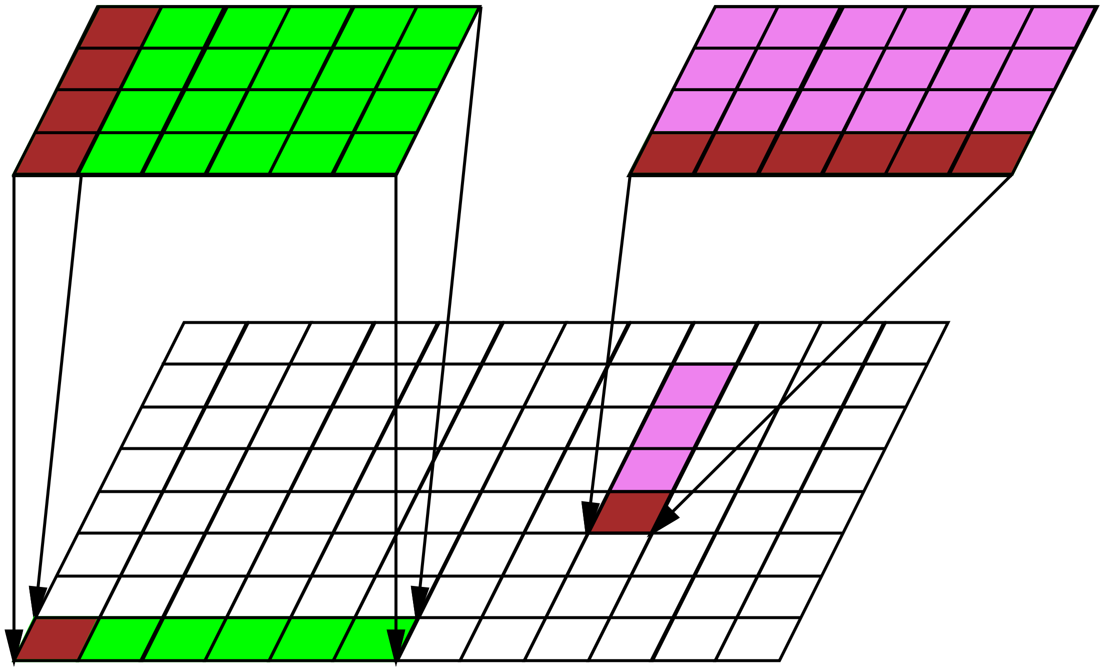
```
align A(i,j) with T(i,5-j)
align B(:,:) with T(11:6:-1,6:3:-1)
```

Transposition



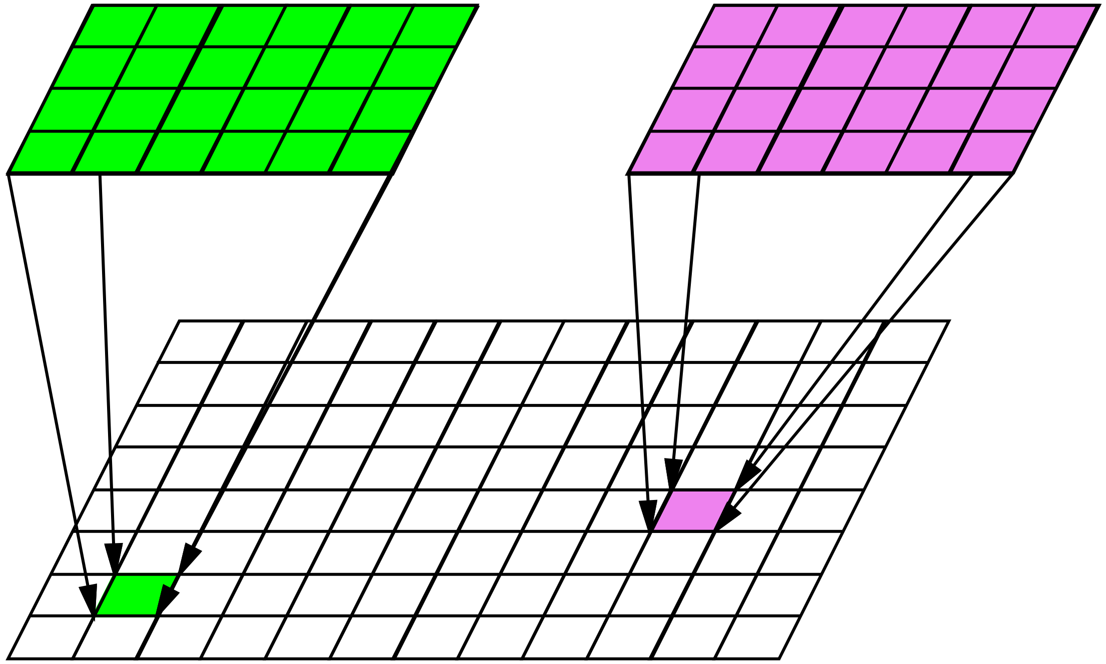
```
align A(i,j) with T(j,i)
align B(i,j) with T(j+3,i+1)
```

Écrasement



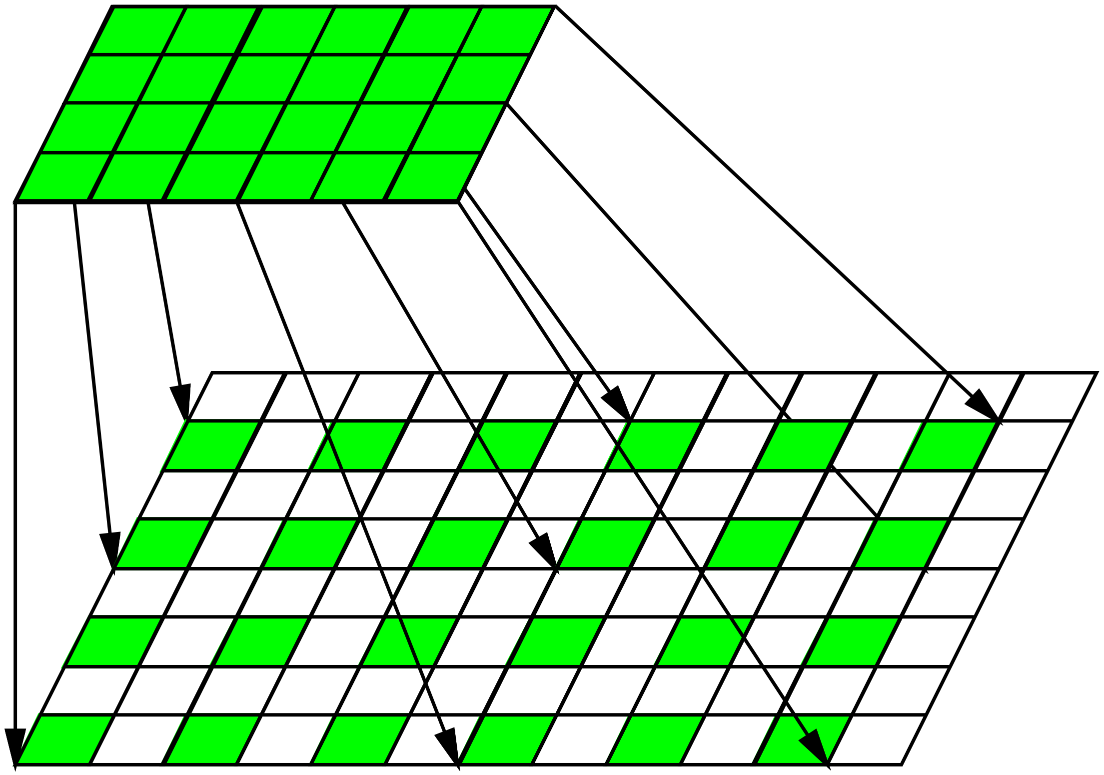
```
align A(i,*) with T(i,1)
align B(*,:) with T(9,4:7)
```

Super écrasement



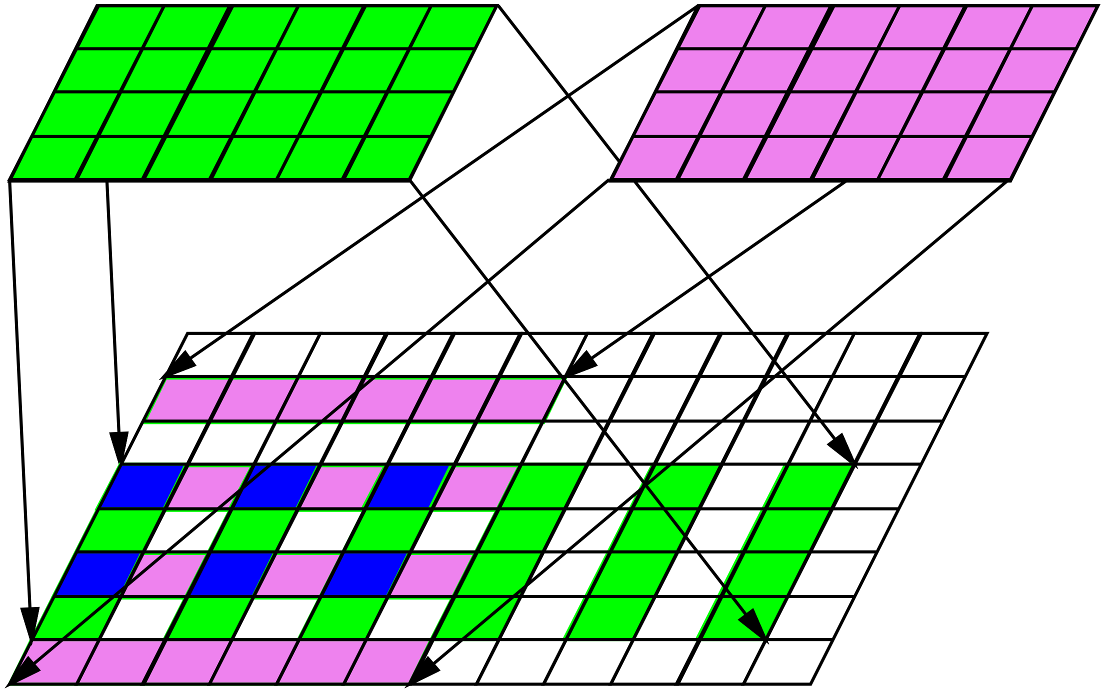
```
align A(*,*) with T(2,2)
align B(*,*) with T(10,4)
```

Éclatement



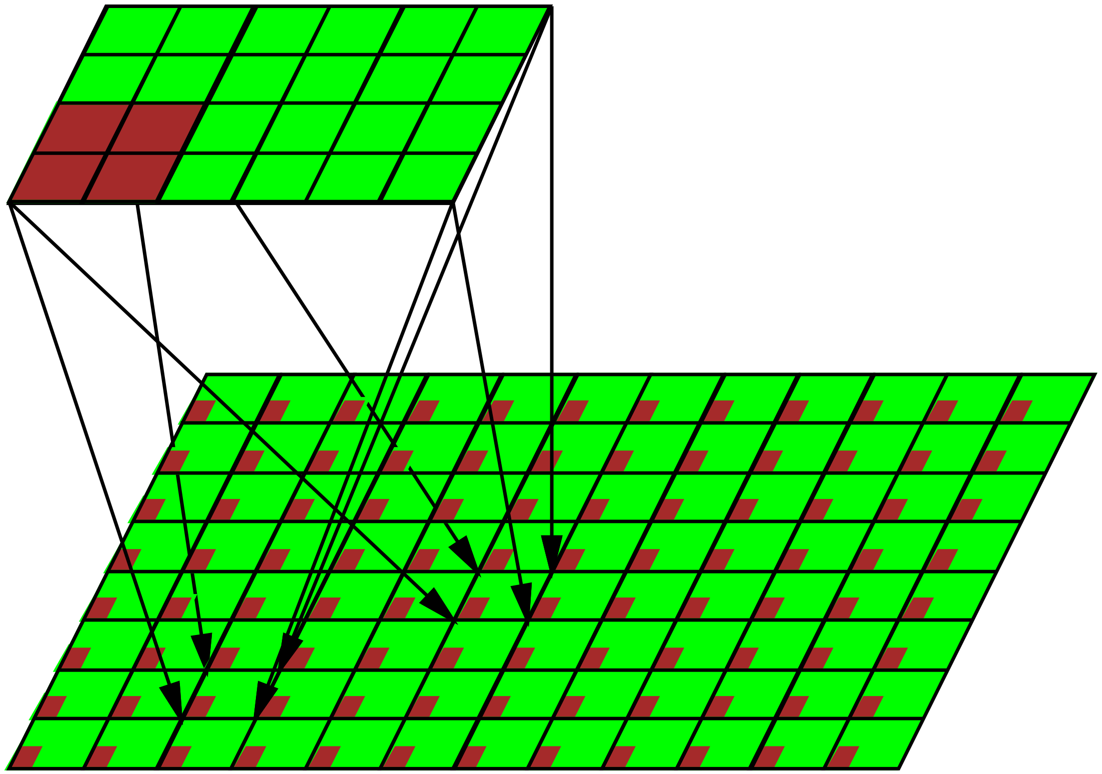
```
align A(i,j) with T(2*i-1,2*j-1)
```

Éclatement...



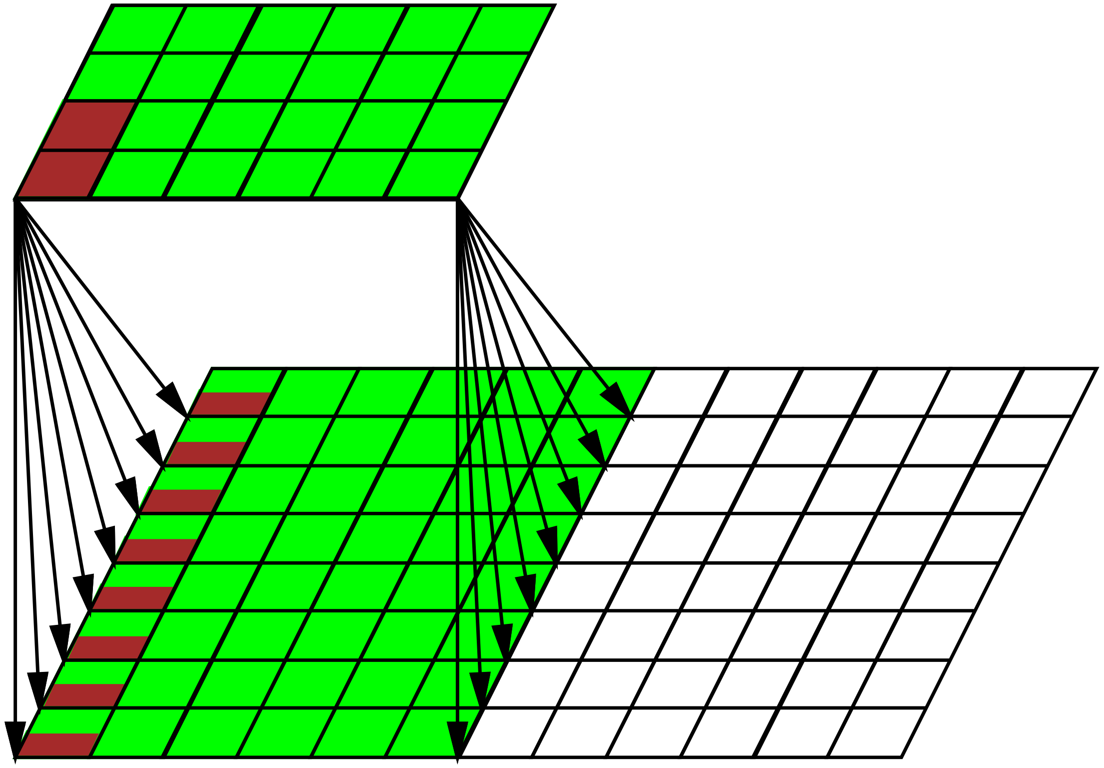
```
align A(i,j) with T(2*i-1,j+1)
align B(i,j) with T(i,2*j-1)
```

« Réplication »



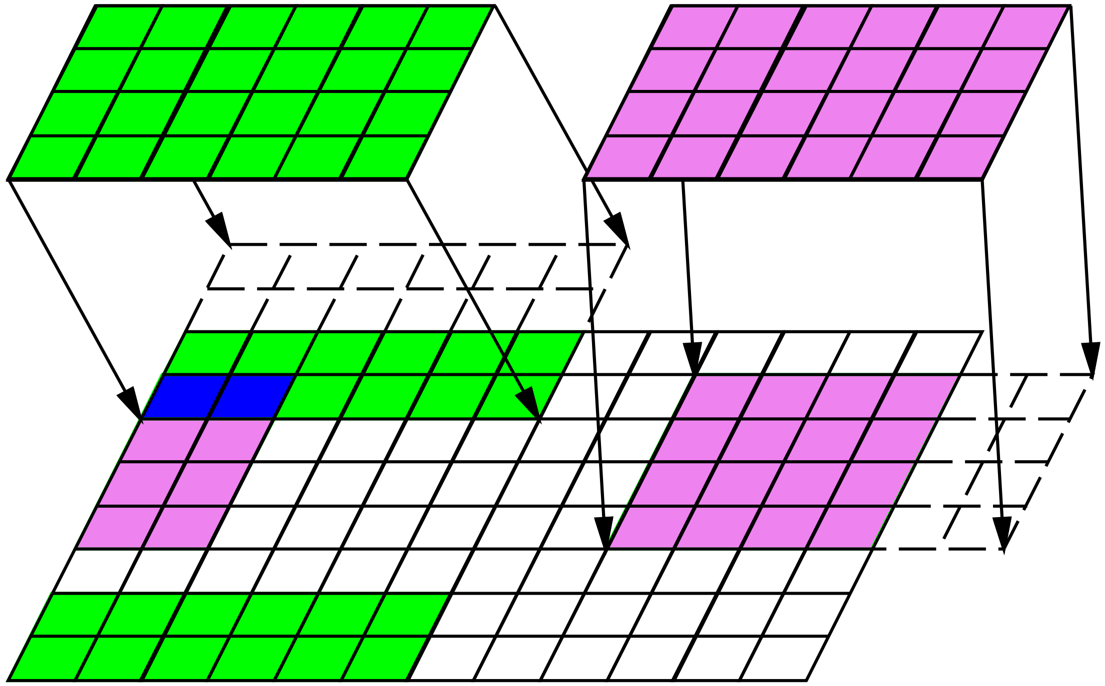
```
align A(*,*) with T(*,*)
```

« Réplication » partielle



```
align A(:,*) with T(:,*)
```

Débordements interdis



```
align A(i,j) with T(i,j+6)   ! non
align B(:,:) with T(9:,4:7)  ! non
```

Processors

`PROCESSORS` est une directive de déclaration

- les *processors* sont des processeurs abstraits
- mais garantie de fonctionnement ssi

  processors = processeurs réels

```
CHPF$ PROCESSORS P1, P2(10,10)
```

Distribution

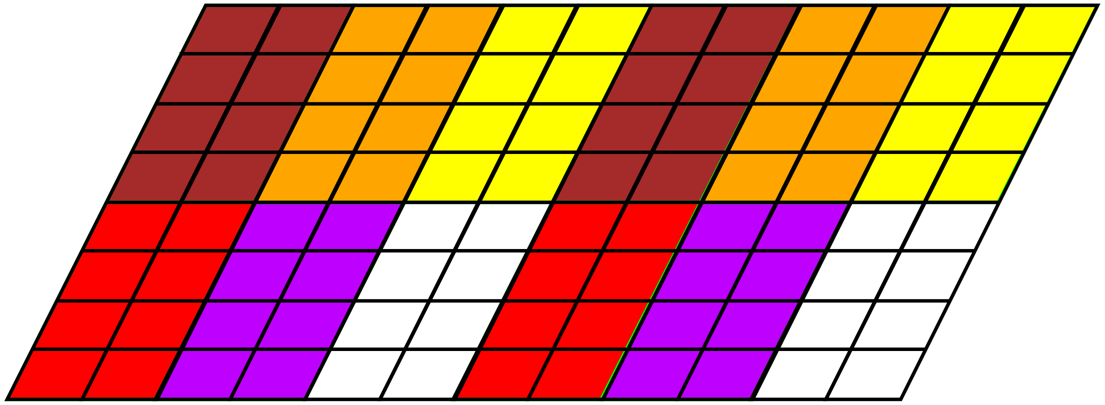
`DISTRIBUTE` est une directive de déclaration

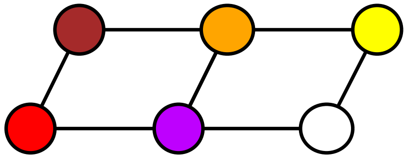
– les *templates* sont distribués sur des *processors*

– on regroupe les éléments de *template* à placer sur un même élément de *processors*

```
CHPF$ DISTRIBUTE T(BLOCK,CYCLIC...) ONTO P1
```

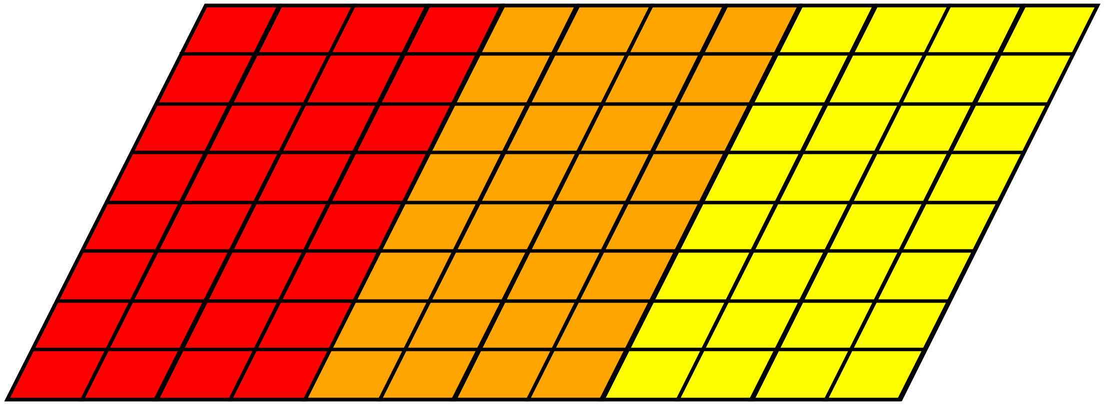
Distribution

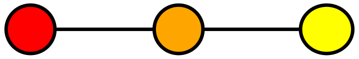




`distribute T(...,...) onto P`

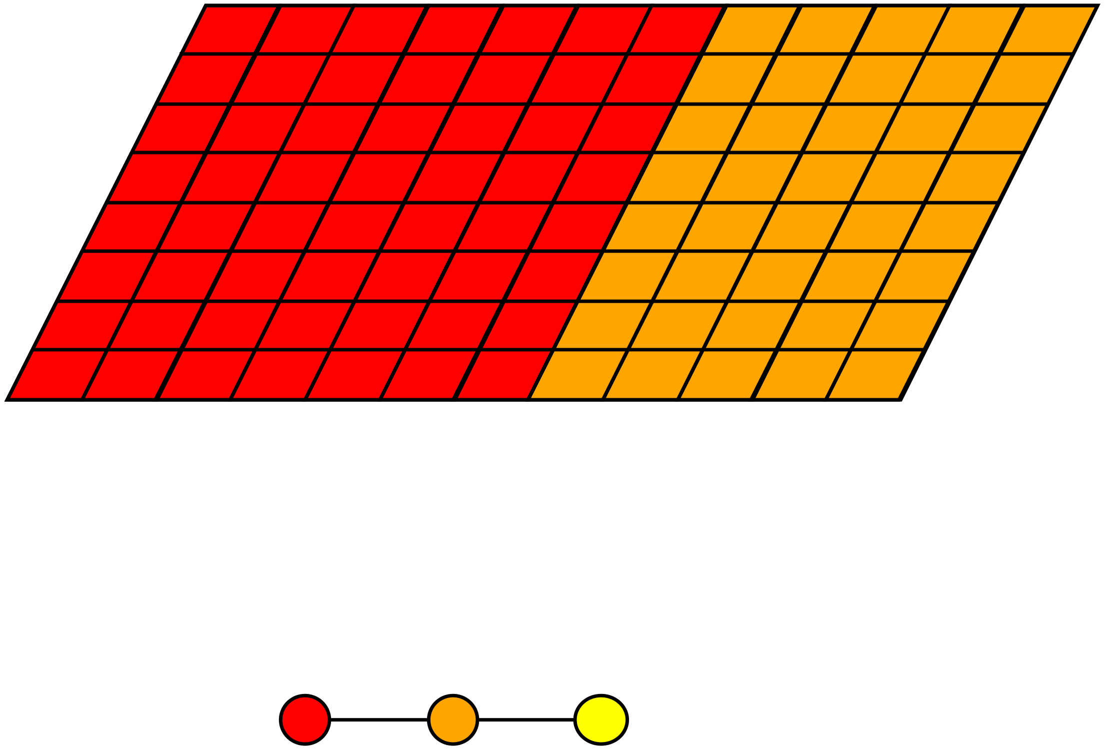
Distribution par `block`





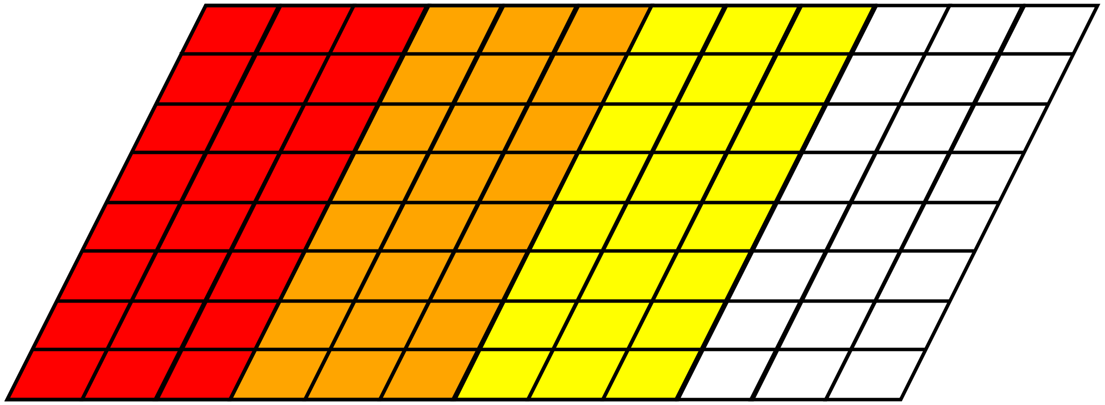
`distribute T(block,*) onto P`

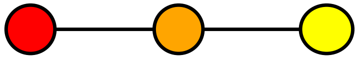
Distribution par `block(n)`



`distribute T(block(7),*) onto P`

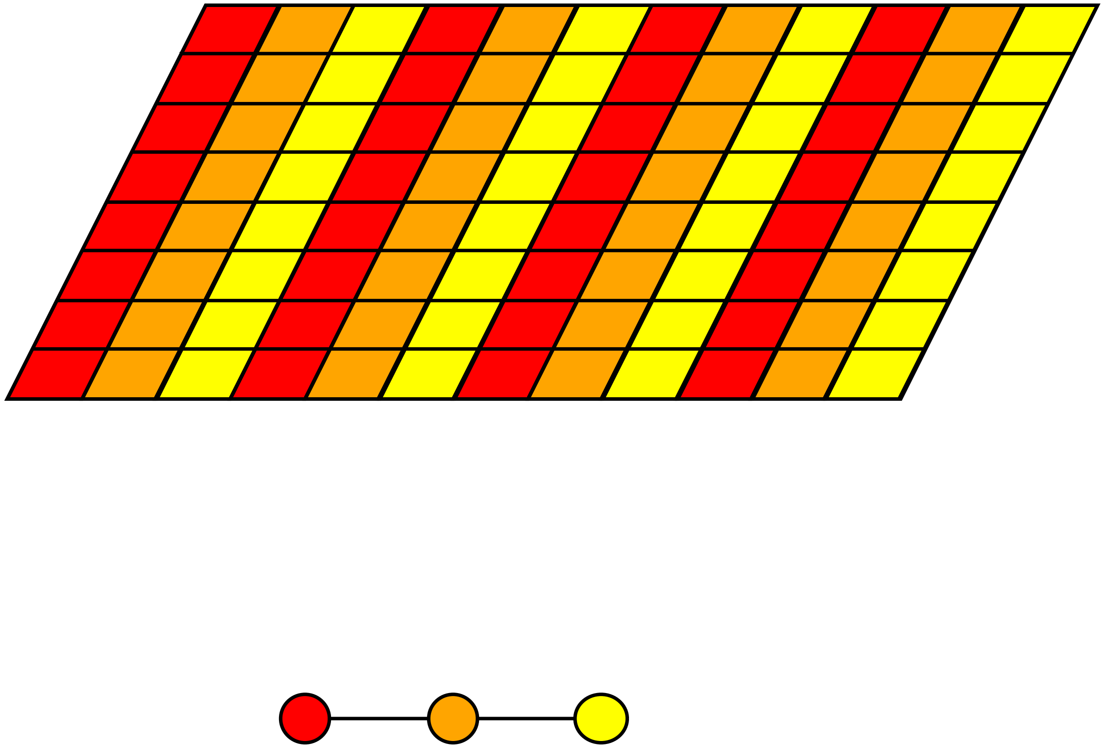
Distribution par `block(n)`





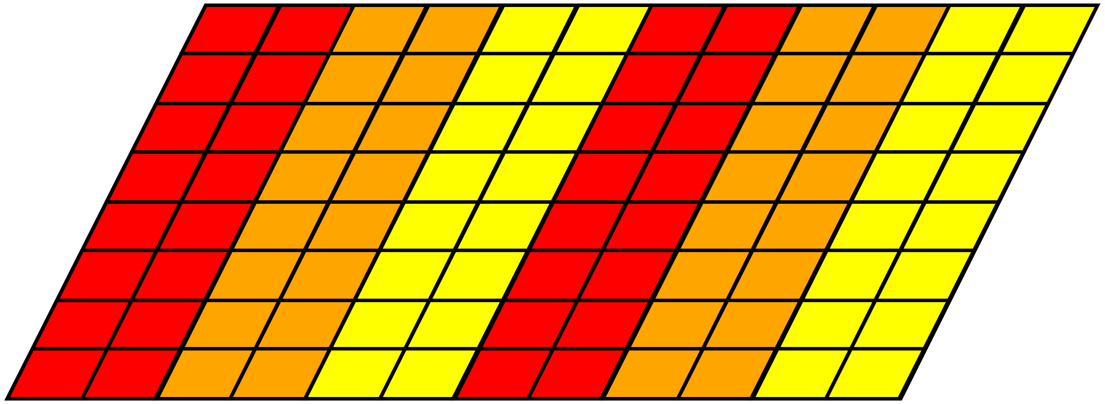
```
distribute T(block(3),*) onto P  ! non
```

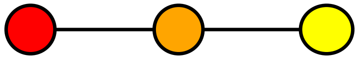
Distribution `cyclic`



`distribute T(cyclic,*) onto P`

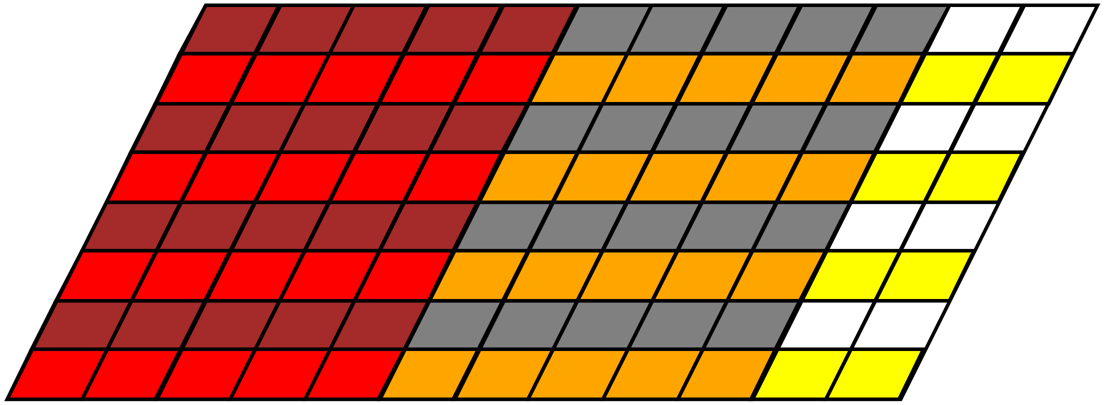
Distribution `cyclic(n)`

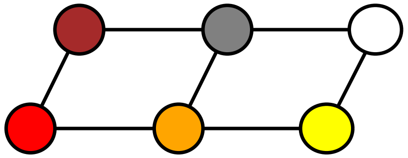




```
distribute T(cyclic(2),*) onto P
```

Distributions multidimensionnelles





```
distribute T(block(5),cyclic(1)) onto P
```

Intérêt des `BLOCK` et `CYCLIC`

- `BLOCK` : répartition régulière pour minimiser les communications dans des calculs à accès voisins

- `BLOCK(n)` : optimisation de la taille des blocks

- `CYCLIC` : répartition de charges lors d'accès non homogènes (matrices triangulaires...)

- `CYCLIC(n)` : répartition de charge pour des accès locaux et non homogènes

Autres Directives

- `REALIGN` : exécutable, alignement dynamique
- `REDISTRIBUTE` : exécutable, redistribution dynamique
- `DYNAMIC` : attribut d'un tableau qui peut être réaligné ou redistribué

pas dans Subset HPF

Remarques

- sucre syntaxique (Fortran 90)
- utilisation d'intrinsics dans les directives
- alignements, distributions et déclaration par défaut
- bizarreries permises
- limitation du placement en HPF

Sucre Syntaxique

```
chpf$ processors, dimension(10) :: PROC
chpf$ template, &
chpf$   distribute(block) onto PROC, &
chpf$   dimension(100), &
chpf$   dynamic :: T1,T2
chpf$ dynamic, align (:) with T1(:) :: A, B
```

Utilisation d'intrinsics

- NUMBER_OF_PROCESSORS() : nombre de processeurs physiques dont dispose l'application à l'exécution.

- PROCESSORS_SHAPE() : dimension de l'arrangement de processeurs physiques.

```
CHPF$ PROCESSORS PROC(NUMBER_OF_PROCESSORS()), &
CHPF$   ARRAYPROC(NUMBER_OF_PROCESSORS()/10,10)
```

Valeurs par défaut

- Alignement à l'identité
- Processors vecteur des processeurs physiques
- Distribution par blocks

Limitations des Placement en HPF

pas de placement très irréguliers :

```
c Vienna Fortran
      ALIGN A(I) WITH B(f(I))
```

- `f` fonction arbitraire
- limite les optimisations...

Placement des arguments d'une sous-routine

- attendu particulier (erreur sinon)
- placement à imposer (dynamiquement)
- prendre le placement comme il vient

Placement à imposer

– comme déclarations statiques

```
      subroutine CALCUL(A)
      real A(100,100)
chpf$ distribute A(block,block)
      ...
```

– redistribution en entrée et sortie

– coup à l'exécution

Placement assuré

- assertion au compilateur
- favorise les optimisations
- vrai pour tous les appels

```
chpf$ align A with * T
chpf$ distribute * T(block)
```

Placement à accepter

- pas d'information... pas d'optimisations ?
- code à générer pour tout placement
- héritage du *template* initial...
- syntaxes

```
      subroutine CALCUL(A)
      real A(100,100)
chpf$ distribute A * onto *
chpf$ inherit A
```

Conclusion, Placement dans HPF

- HPF proche de Fortran D pour le placement
- permet la duplication
- adapté aux tableaux
- limitations pour faciliter les optimisations
- directives = conseils

3 Boucles en HPF

Groupe de Charles Koelbel, Rice U.

– boucle séquentielle `DO`

– boucle *dataparallèle* `FORALL`

```
a(:,:) = a(:,:) + b(:,:)
```

```
forall(i=2:n) a(i)=a(i-1)
```

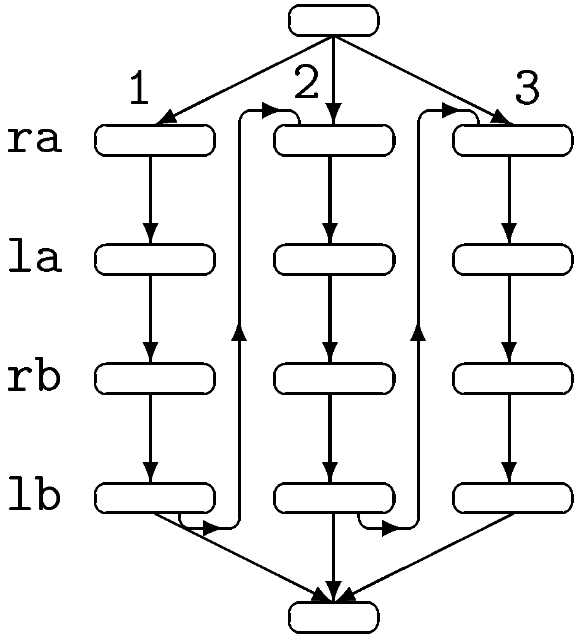
– boucle parallèle, directive `INDEPENDENT`

itérations indépendantes les unes des autres

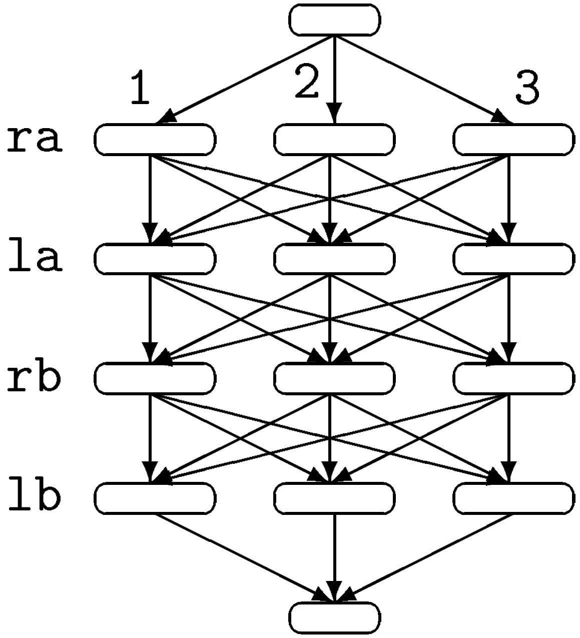
```
chpf$ independent
      do 10 i=1,n
 10      a(i) = b(p(i)) ! p is a permutation
```

DO et FORALL

```
DO i = 1, 3
  lhsa(i) = rhsa(i)
  lhsb(i) = rhsb(i)
END DO
```

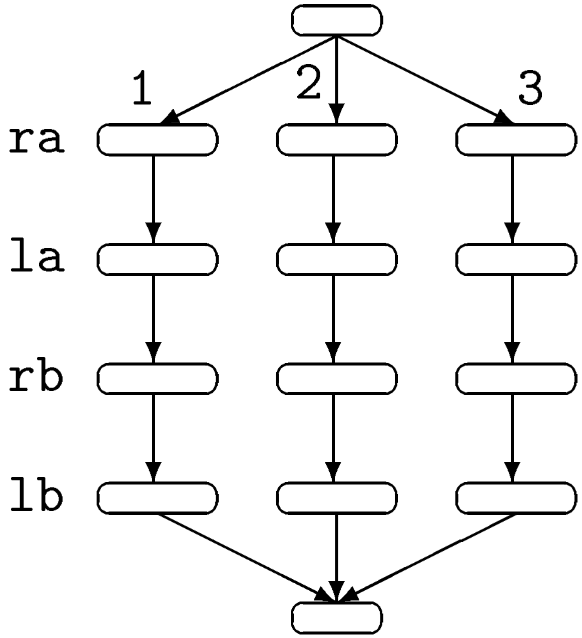



```
FORALL ( i = 1:3 )
  lhsa(i) = rhsa(i)
  lhsb(i) = rhsb(i)
END FORALL
```

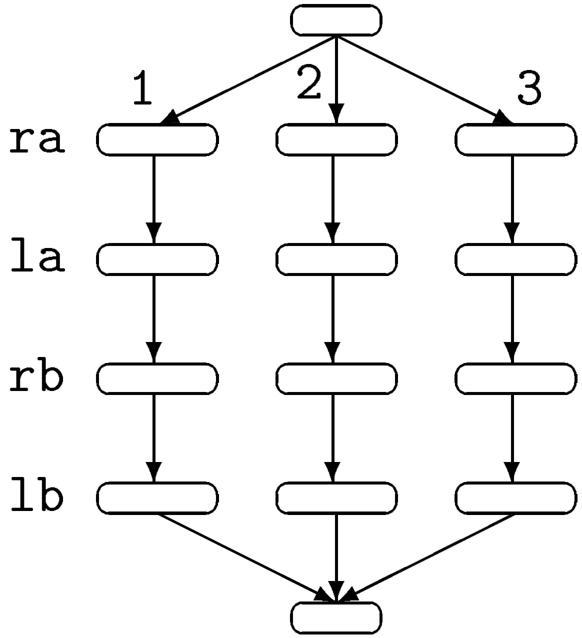


INDEPENDENT

```
!HPF$ INDEPENDENT
DO i = 1, 3
  lhsa(i) = rhsa(i)
  lhsb(i) = rhsb(i)
END DO
```




```
!HPF$ INDEPENDENT
FORALL( i = 1:3 )
  lhsa(i) = rhsa(i)
  lhsb(i) = rhsb(i)
END DO
```



FORALL

boucles dataparallèles à **3** saveurs :

- **implicite** : sections de tableaux (Fortran 90)

```
A = B + C           ! matrices
D(1:20) = D(2:21) ! shift
```

- généralisation, `FORALL` **statement**

- extension, `FORALL` **construct**

`FORALL` **statement**

le `FORALL` statement généralise les sections de tableaux :

- sémantique dataparallèle (SIMD)
- une seule affectation
- masquage
- appel à des procédures `PURE` (intrinsics...)

```
FORALL(i=1:n, ..., mask) affectation
```

Exemples de `FORALL` statements

```
FORALL (i=1:n, j=1:n) x(i,j) = y(j,i)

FORALL (i=1:n, y(i).NE.0.0)
  x(i) = 1.0 / y(i)

FORALL (i=2:n-1)
  x(i) = 0.5*x(i) + 0.25*(x(i-1)+x(i+1))

FORALL (i=1:n) x(i) = SQRT(y(i))
```

FORALL **construct**

extensions du FORALL statement pour :

- plusieurs affectations
- corps peut contenir un WHERE...

```
FORALL (i=1:n, ... , mask)
  WHERE ou FORALL ou affectations...
END FORALL
```

Exemples de `FORALL` constructs

initialisation d'une matrice carrée à l'identité :

```
FORALL(i=1:n, j=1:n)
  WHERE (i.EQ.j)
    a(i,j) = 1.0  ! diagonale
  ELSEWHERE
    a(i,j) = 0.0  ! reste
  END WHERE
END FORALL
```

affectations multiples, avec appel à des intrinsics :

```
FORALL(i=1:n, j=1:m)
  a(i,j) = 0.0
  b(i,j) = 1.0
  c(i,j) = 1.0 / (i+j+1)  !
  d(i,j) = SQRT(c(i,j))   ! dependence
END FORALL
```

Attribut `PURE`

fonctions sans effets de bords :

- E/S, `STOP`, common et paramètres
- fonctions intrinsèques

$\rightarrow$ fonctions pures, boucles parallèles

2 déclarations : interface et fonction

nombreuses contraintes : contrôle $>$ flexibilité...

directive `INDEPENDENT`

cette directive désigne les boucles parallèles :

- parfaitement emboîtées
- itérations indépendantes (entremélée. . .)
- variables privées (propres à chaque itération)

```
CHPF$  INDEPENDENT(i,...), NEW(var-list)
       DO i=1,n ... / FORALL (i=1:n) ...
```

Exemples de la directive INDEPENDENT

```
CHPF$  INDEPENDENT
       DO i=1,100
         A(P(i)) = B(i)  ! P permutation
       END DO

CHPF$  INDEPENDENT
       FORALL (i=1:n)
         A(P(i)) = B(i)  ! idem
       END DO
```

multiplication de deux matrices carrées :

```
CHPF$  INDEPENDENT(i,j), NEW(TEMP)
       DO 10 i=1,n
         DO 10 j=1,n
           TEMP = 0.0
           DO 20 k=1,n
 20          TEMP = TEMP + A(i,k)*B(k,j)
           C(i,j) = TEMP
 10    CONTINUE
```

Conclusion boucles parallèles

- dépassement des simples directives
- pas de `ON`
- sémantique du `FORALL` inhabituelle
- Subset : pas de `FORALL` construct

Réductions

- Fortran 90 : `SUM MAXVAL...`
- HPF : `MAXLOC...`

Reste

- **E-S**, Bob Knighten (Intel)
- **Fortran 90**, Mary Zosel (LLNL)
- **intrinsics**, Rob Schreiber (RIACS)
- **extrinsics**, Marc Snir (IBM)
- **subset**, Mary Zosel, (LLNL), David Loveman (DEC)
- **benchmark**, Jeoffrey Fox, ?

Entrées-Sorties

$\boxed{\text{Rien}}$

Fortran 90

$\boxed{\text{Tout, ISO IEC 1539 : 1991 (E)}}$

Sauf quelques restrictions concernant les

« Associations de Stockage »

des tableaux répartis.

Common, Equivalence, Sequence

– F77 : données `common` partagées entre modules

```
common /foo/ a(10,10), b(100)
```

  – renommage :

```
common /foo/ new_a(10,10), new_b(100)
```

  – déformation :

```
common /foo/ a(100), b(20,5)
```

  – découpage :

```
common /foo/ c(50), d(100), e(50)
```

  – couverture :

```
common /foo/ cover(200)
```

- compatibilité placement ?
  - réarrangements trop coûteux
  - non garantie $\rightarrow$ restrictions
  - 2 directives pour le compilateur

    ```
    chpf$ sequence /foo/
    chpf$ non sequence /bar/
    ```

  - `common` séquentiel (non distribuable)
  - `common` non séquentiel (distribuable)

Equivalence

superposition de variables

```
  common /bar/ a(10), b(10), c(2,5), d(10)
  real e(10), f(20), o(40)
  equivalence(a(1),e(1)) ! renomme
  equivalence(f(1),b(1)) ! regroupe...
  equivalence(o(1),a(1)) ! couverture
! distribute o(cyclic(2))
```

– une couverture 1D peut être distribuée. . .

Fonctions intrinsèques

– enquête sur le **système**

– enquête sur le **placement**

– nouvelles fonctions

– `HPF_LIBRARY`

Système

- NUMBER_OF_PROCESSORS([DIM])
- PROCESSORS_SHAPE()

  retourne un vecteur. . .

Placement

- HPF_ALIGNMENT()
- HPF_TEMPLATE()
- HPF_DISTRIBUTION()

Fonctions

– de Fortran 90 : `SUM`, `MATMUL`

– élémentaire `ILEN` *(integer length)*

– extension de `MAXLOC` et `MINLOC`

`DIM` optionel, comme `MAXVAL` et `MINVAL`

fonction de localisation

Réductions

– Fortran 90 : masquables, par dimensions

  – `ALL` : et logique

  – `ANY` : ou logique

  – `COUNT` : nombre de V

  – `MAXVAL`, `MINVAL`

  – `PRODUCT`, `SUM` : (types. . .)

- HPF :
  - `PARITY` : ou exclusif
  - `IALL`, `IANY`, `IPARITY`

    versions bit à bit sur entiers.
- Versions parallèles
  - tableaux distribués

    dimensions, types de distribution
  - exploitation de la machine
  - gestion du masque

XXX_SCATTER

réductions généralisées...

- tableau source
- base de départ (conforme au résultat)
- tableaux des indexes contribuants...

  autant que de dimensions de B

  conforme à la source

  index dans le résultat
- masquage facultatif

```
A   = [ 10 20 30 40 -10 ]

IDX = [  3  2  2  1   1 ]

B   = [ 1  2  3  4 ]

SUM_SCATTER(A, B, IDX, MASK=(A.GT.0))

->  [ 41 52 13  4 ]
```

XXX_PREFIX XXX_SUFFIX

- opérations cumulées
- tableau linéarisé. . .
- propagation avant et arrière
- dimension, masquage, segment, exclusive

  `DIM MASK SEGMENT EXCLUSIVE`

Exemple SUM

$$\texttt{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \texttt{Z} = \begin{vmatrix} \texttt{T} & \texttt{T} & \texttt{F} \\ \texttt{T} & \texttt{F} & \texttt{F} \\ \texttt{T} & \texttt{T} & \texttt{F} \end{vmatrix}$$

$$\texttt{SUM\_PREFIX(A)} = \begin{pmatrix} 1 & 14 & 30 \\ 5 & 19 & 36 \\ 12 & 27 & 45 \end{pmatrix}$$

$$\texttt{SUM\_SUFFIX(A)} = \begin{pmatrix} 45 & 33 & 18 \\ 44 & 31 & 15 \\ 40 & 26 & 9 \end{pmatrix}$$

$$\texttt{SUM\_PREFIX(A, DIM=2)} = \begin{pmatrix} 1 & 3 & 6 \\ 4 & 9 & 15 \\ 7 & 15 & 24 \end{pmatrix}$$

$$\texttt{SUM\_PREFIX(A, MASK=Z)} = \begin{pmatrix} 1 & 14 & 22 \\ 5 & 14 & 22 \\ 12 & 22 & 22 \end{pmatrix}$$

$$\texttt{SUM\_PREFIX(A, SEGMENT=Z)} = \begin{pmatrix} 1 & 14 & 3 \\ 5 & 5 & 9 \\ 12 & 8 & 18 \end{pmatrix}$$

```
SUM_PREFIX(A, DIM=2, EXCLUSIVE=.TRUE.) =
```

$$\begin{pmatrix} 0 & 1 & 3 \\ 0 & 4 & 9 \\ 0 & 7 & 15 \end{pmatrix}$$

```
SUM_PREFIX(A, DIM=2, SEGMENT=Z, MASK=A.GT.4) =
```

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 5 & 11 \\ 7 & 15 & 9 \end{pmatrix}$$

HPF_LIBRARY

- réductions ajoutées
- autres fonctions élémentaires :
  - LEADZ : *leading zeros*
  - POPCNT : nombre de bits à 1
  - POPPAR : parité d'un entier
- tri
  - GRADE_DOWN : permutation ascendante
  - GRADE_UP : descendante

Exemple GRADE_UP

$$\texttt{A} = \begin{pmatrix} -1 & 5 & -3 & 4 \\ 2 & 1 & 0 & 9 \end{pmatrix}$$

$$\texttt{GRADE\_UP(A)} = \begin{pmatrix} 1 & 3 \\ 1 & 1 \\ 2 & 3 \\ 2 & 2 \\ 2 & 1 \\ 1 & 4 \\ 1 & 2 \\ 2 & 4 \end{pmatrix}$$

Extrinsics Procedures

possibilité d'interfaçage avec

- d'autres langages (C...)
- d'autres styles de programmation (messages...)
- librairies optimisées...

syntaxe, définition de la vue de la procédure pour HPF :

```
INTERFACE
  EXTRINSIC (HPF_LOCAL) SUBROUTINE FOO(BAR)
  REAL BAR(:)
  !HPF$ DISTRIBUTE BAR(BLOCK)
  END SUBROUTINE
END INTERFACE
```

avec HPF_LOCAL : type de langage concerné

Subset

HPF, c'est :

- Fortran 90 avec quelques restrictions
- directives de placement et de parallélisme
- 2 foralls
- instrinsics, librairie, etc.

Subset HPF officiel : disponibilité des compilateurs le **Subset HPF**, c'est Fortran 77 (restreint) et :

- de Fortran 90 :
  - `DO WHILE`, qq intrinsics, réductions, interfaces
  - sections de tableaux, `WHERE`...
  - `ALLOCATE`...
- **HPF** sauf
  - directives **dynamiques**
  - `FORALL` construct
  - `HPF_LIBRARY`, extrinsics...

Benchmark HPF

HPF/Fortran-D Benchmarking Suite, Release 2.01

- tests des compilateurs?
  - **validation** des compilateurs
  - **évaluation** des performances
- pour l'instant, ensemble de programmes
  - Fortran 77, Fortran 77 + messages
  - Fortran 90, Fortran 90 parallel, Fortran 90-D

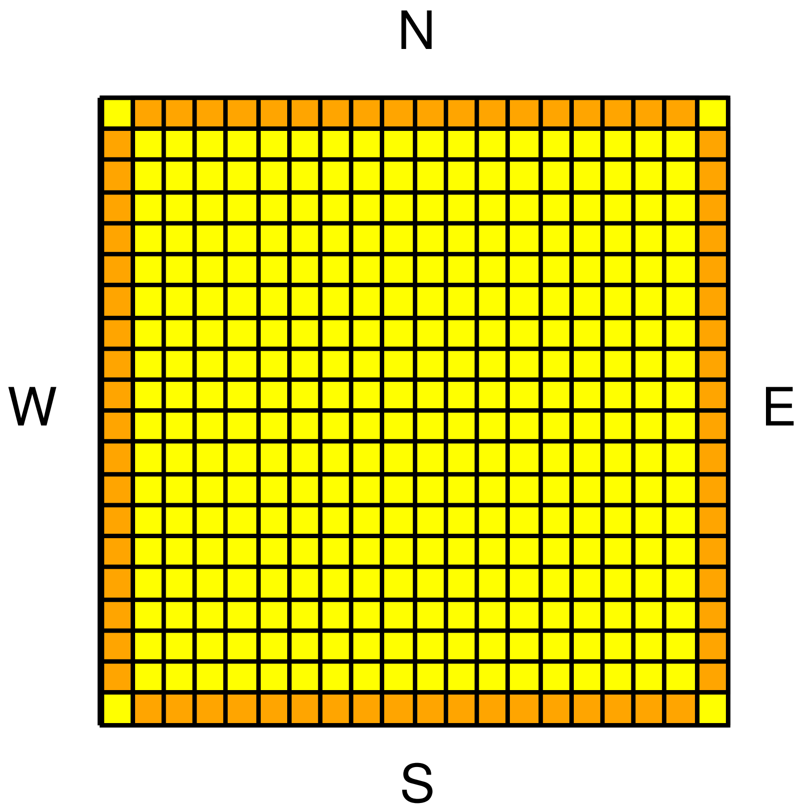
produit de matrices

```
       REAL,DIMENSION(100,100) :: A,B,C
CHPF$  ALIGN A(I,*) WITH C(I,*)
CHPF$  ALIGN B(*,J) WITH C(*,J)
CHPF$  INDEPENDENT(i,j), NEW(temp)
       DO 10 i=1,100
         DO 10 j=1,100
           temp = 0.0
           DO 20 k=1,100
 20          temp = temp + A(i,k)*B(k,j)
           C(i,j) = temp
 10    CONTINUE
```

Thermique, déclarations

```
C équilibre thermique d'une plaque homogène,
c   avec conditions aux limites.
      PROGRAM THERMIQUE
      PARAMETER(N=100)
Champs de temperature TC courante, TS suivante
      REAL, DIMENSION(N,N) :: TS,TC
Conditions aux limites sur les 4 bords
C N = Nord ; S = Sud ; W = Ouest ; E = Est
      REAL, DIMENSION(2:N-1) :: N,S,W,E
```

Vue de la plaque



directives

```
C alignement des vecteurs N, S, W, E
CHPF$ ALIGN N(:) WITH TC(1,2:)
CHPF$ ALIGN S(:) WITH TC(N,2:)
CHPF$ ALIGN W(:) WITH TC(2:,1)
CHPF$ ALIGN E(:) WITH TC(2:,N)
C alignement du second tableau
CHPF$ ALIGN TS(:,:) WITH TC(:,:)
C distribution sur une grille
CHPF$ DISTRIBUTE TC(BLOCK,BLOCK)
```

initialisations

```
C Entree des donnees
      INPUT N(:),S(:),W(:),E(:)
C initialisation TC
      TC(:,:)  = 300
      TC(1,2:) = N(:)
      TC(N,2:) = S(:)
      TC(2:,1) = W(:)
      TC(2:,N) = E(:)
```

algorithme

```
C initialisation TS, premiere iteration
      CALL ITERE(TS,TC)
C iterations jusque convergence
      DO WHILE (MAXVAL(ABS(TS-TC)).GT.epsilon)
          TC(:,:) = TS(:,:)
          CALL ITERE(TS,TC)
      END DO
C sortie du resultat
      PRINT *, TS(:,:)
      END PROGRAM
```

itération

Calcul d'une iteration

```
      SUBROUTINE ITERE(T2,T1)
      REAL, DIMENSION(N,N) :: T1,T2
CHPF$ DISTRIBUTE *(BLOCK,BLOCK) :: T1,T2
      T2(2:N-1,2:N-1) =                          &
          0.25 * (T1(1:,2:) + T1(2:,1:) +    &
                  T1(3:,2:) + T1(2:,3:))
      END
```

Adéquation de HPF

- accès locaux, distribution par `BLOCK`.
- sections de tableau (F90)
- `WHILE`...
- bonne adaptation des directives au problème
- algorithme sans particularités aux bords...
- compilation ?

Situation actuelle

- **nombreuses annonces** : DEC, Intel, MasPar, TMC, Meiko, IBM, Fujitsu, HP, Cray, Lahey, Archipel...
- **HPC** en cours de spécification
- second Forum pour HPF 2 depuis janvier 94
  - I/O
  - placement irréguliers
  - parallélisme de tâche

HPF-2

– réunions commencées dès 1994

– décisions reportées en 1995. . . puis 1996

– document de travail et de réflexion

HPF-2 Scope of Activities and Motivating Applications

– s'éloigne parfois du langage

Contenu

- extension du placement
- parallélisme tâche
- entrées-sorties
- aide aux communications. . .
- environnement du langage
- kernel HPF. . .

Extension du placement

- irrégulier (fonctions, tableaux. . .)
- structures dynamiques (arbres)
- types dérivés de F90

  peuvent contenir des tablaux
- sous-ensemble de processeurs

  sections. . . `P(1:N), ONTO P(1:5)`
- vues de processeurs
- . . .

Parallélisme de tâche, Contrôle des calculs

- distribution des itérations EXECUTE_ON

  $\rightarrow$ *contrôle répartition des calculs*

- réductions dans INDEPENDENT

  REDUCTION

- sections critiques

  CRITICAL

- DOACROSS (dépendances gérées par synchro)

- processus multiples $\rightarrow$ communications...
  - passage de messages
  - canaux/fichiers virtuels
  - données partagées
- sections parallèles

```
!hpf$ begin parallel
      call F1
      ...
      call Fn
!hpf$ end parallel
```

Entrées-Sorties

problème difficile, lié au matériel

– transparence. . . (rien ?)

– distribution de fichiers. . .

– *Out-of-core*

– *checkpointing/restart*

– E/S asynchrones

Aide aux communications

- communications régulières

  $\rightarrow$ décallages, réalignement...

- réutilisation d'un schéma de communications
- début de communications avancés
- accès locaux garantis ($\neq$ `INDEPENDENT`)
- vues d'un tableau...

Environnement

– quelques propositions dans HPF-1

– interactions entre langages et outils. . .

kernel HPF

HPF simplifié

- assersion du kernel
- placement statique
- syntaxe simplifiée (?)
- alignements directs

  ni duplication, ni écrasement
- pas d'association *à la Fortran*

kernel HPF

– HPF-1 et *subset* HPF

temps de développements des compilateurs

estimés par le Forum

– HPF-2 et *kernel* HPF...

efficacité des compilateurs

indique l'état de l'art...

HPF-2

- propositions : effet raton-laveur. . .
- kernel inutilement réduit
- analyse d'applications

Références

- High Performance Fortran Language Specification

  3 mai 93

- Quel Fortran pour la Programmation Massivement Parallèle?

  Alexis Platonoff, note CEA 2713, novembre 1992

Langage et compilation

– présentation d'un langage :

  utilisateur : *syntaxe, expressivité, compacité, sémantique*

  $\rightarrow$ SML...

– utilisateur exigeant : *portabilité, réutilisabilité, performance*

  $\rightarrow$ standard, C++, Fortran 77, HPF ?

– compilateur : *simplicité, optimisabilité, taille*

Compilation de HPF

Problèmes

- Fortran 90
- sémantique étendue du placement
- énumérations des itérations locales
- génération des messages
- appels de procédures
- entrées sorties
- fonctions intrinsèques

Fortran 90

ISO/IEC 1539 : 1991 (E)

– 370 pages de spécifications

– contient Fortran 77

– beaucoup de sucre syntaxique

– pointeurs, structures, modules, . . .

trop gros : rejet du DoD

Sémantique du placement

- définitions dynamiques
- gestion des templates
- alignements non unitaires
- distributions cycliques généralisées `cyclic(n)`
- réplication
- réalignements et redistributions dynamique

  dans le programme, aux appels de procédures

Problèmes liés au placement

– allocation des tableaux locaux aux processeurs

  minimisation de l'espace mémoire

  accès efficace

– irrégularités permises par le langage

  alignements et distributions

– connaissance à la compilation...

  paramètres $\rightarrow$ à l'exécution

Exemples d'irrégularités

```
      real a(10)
chpf$ template t(20)
chpf$ align a(i) with t(2*i)
chpf$ distribute t(cyclic(3))
```

groupements par 1 ou 2 éléments

```
      real a(10)
chpf$ template t(40)
chpf$ align a(i) with t(4*i)
chpf$ processors p(4)
chpf$ distribute t(cyclic)
```

a est entièrement sur p(4)

Énumération des itérations locales

– plusieurs « sémantiques », `forall` et `independent`

– enumérations des itérations locales

  owner computes rule sur tableaux répartis

– gardes sur les processeurs non concernés

– efficacité :

  cache : lien avec l'allocation

  minimisation du surcoût de contrôle

Génération des communications

- difficulté : placement + énumération
- efficacité des messages générés
- gestion des temporaires

  allocation, accès, gardes réduites
- exploitation de la machine

  voisinage, broadcast, . . .

Passage de paramètres répartis

- redistributions à l'entrée
- passage de sections de tableau !

  répartition non HPF des sous-tableaux
- appels de contextes différents

  plusieurs compilations ?

  copies dans des temporaires ?

Entrées Sorties

- loin du séquentiel
- données réparties sur les processeurs
- utilisation de la machine

  entrées sorties parallèles
- être capable de recharger les données

  description du stockage

$\boxed{\rightarrow \text{HPF2}}$

Fonctions intrinsèques

– Fortran 90

– extensions de HPF

  réductions, requêtes systèmes, etc.

– très nombreuses

– très paramétrables (types, tailles, dimensions)

Que sait-on faire ?

HPF au delà de l'état de l'art?

- alignements simples (décalages)
- distributions par `block`
- recouvrements au voisinage (Gerndt)
- motifs réguliers de communications
- résolution dynamique des accès aux variables réparties
- cas irréguliers : `PARTI`

Que saura-t-on faire ?

Optimisme

- équations linéaires :

  alignement, distribution, accès

- inéquations linéaires

  déclarations, bornes de boucles

$\boxed{\rightarrow \text{ résolution d'équations et polyèdres}}$

Exemple Chatterjee et al.

```
      array A(0:42)
chpf$ template T(0:127)
chpf$ processors P(0:3)
chpf$ align A(i) with T(3*i)
chpf$ distribute T(cyclic(4)) onto P
      A(0:42:3)
```

Template `T(0:127)`

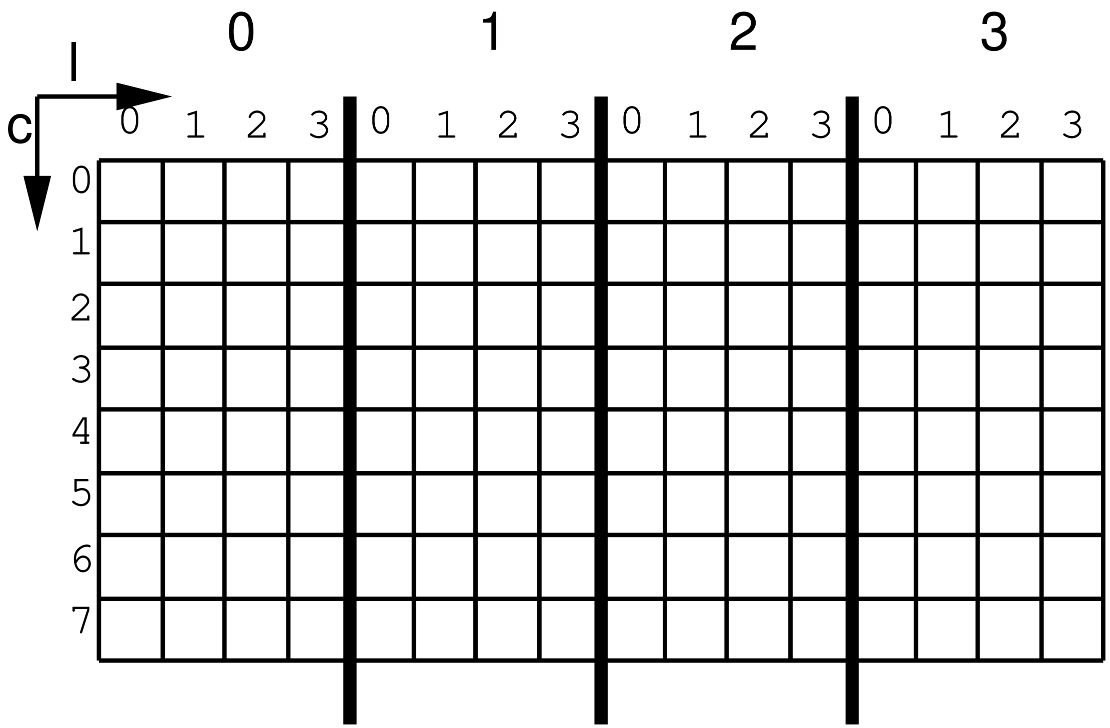
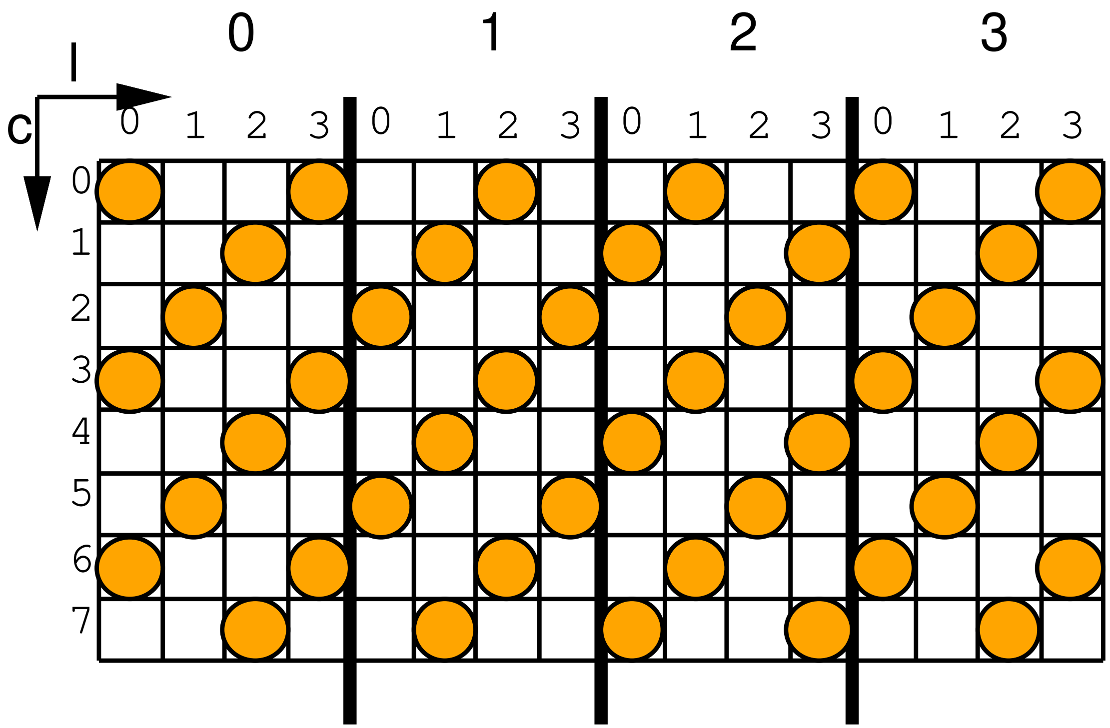
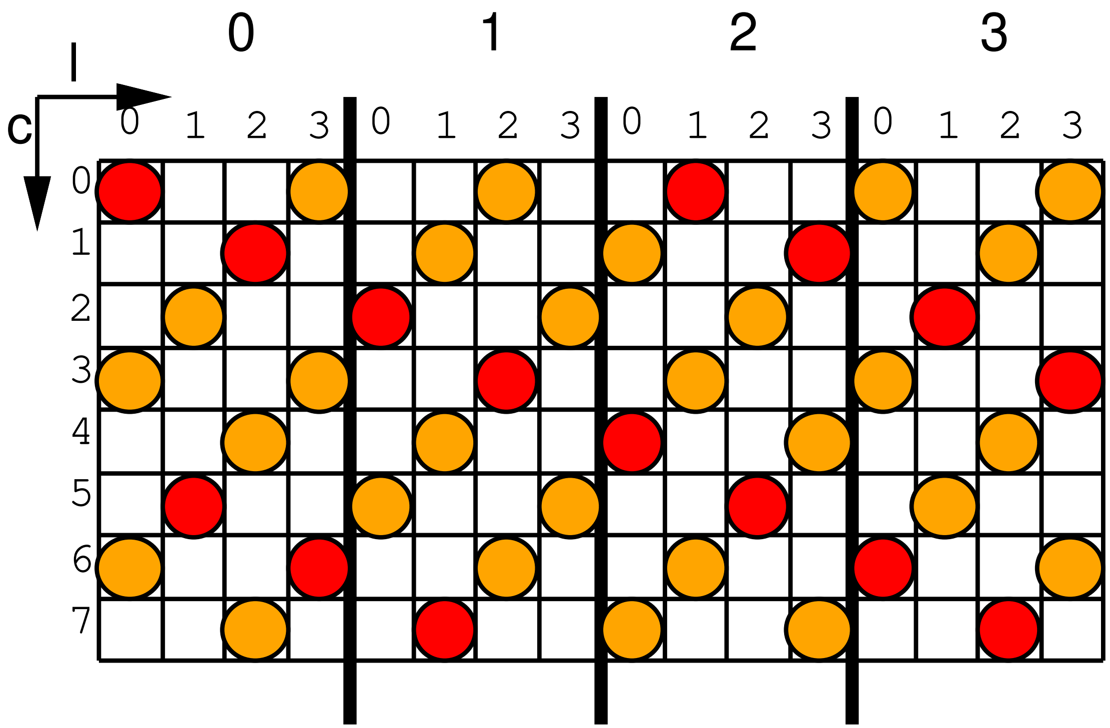


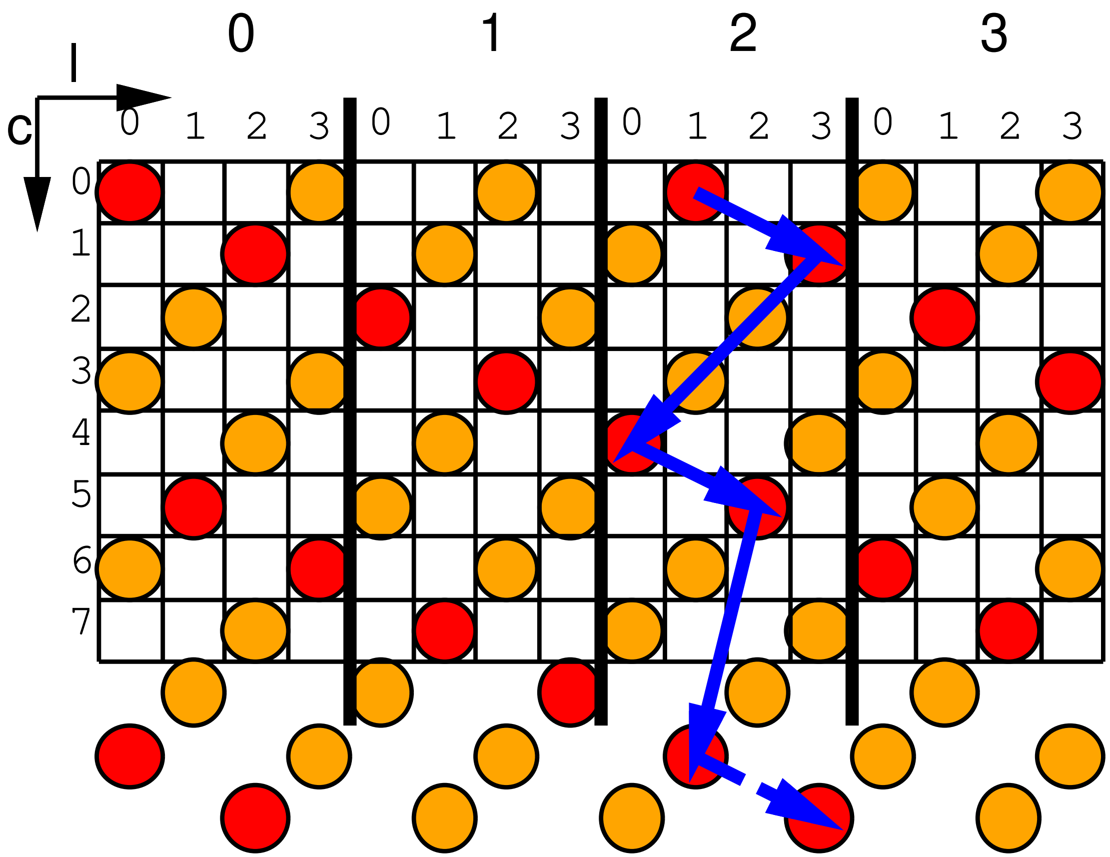
Tableau `A(0:42)`



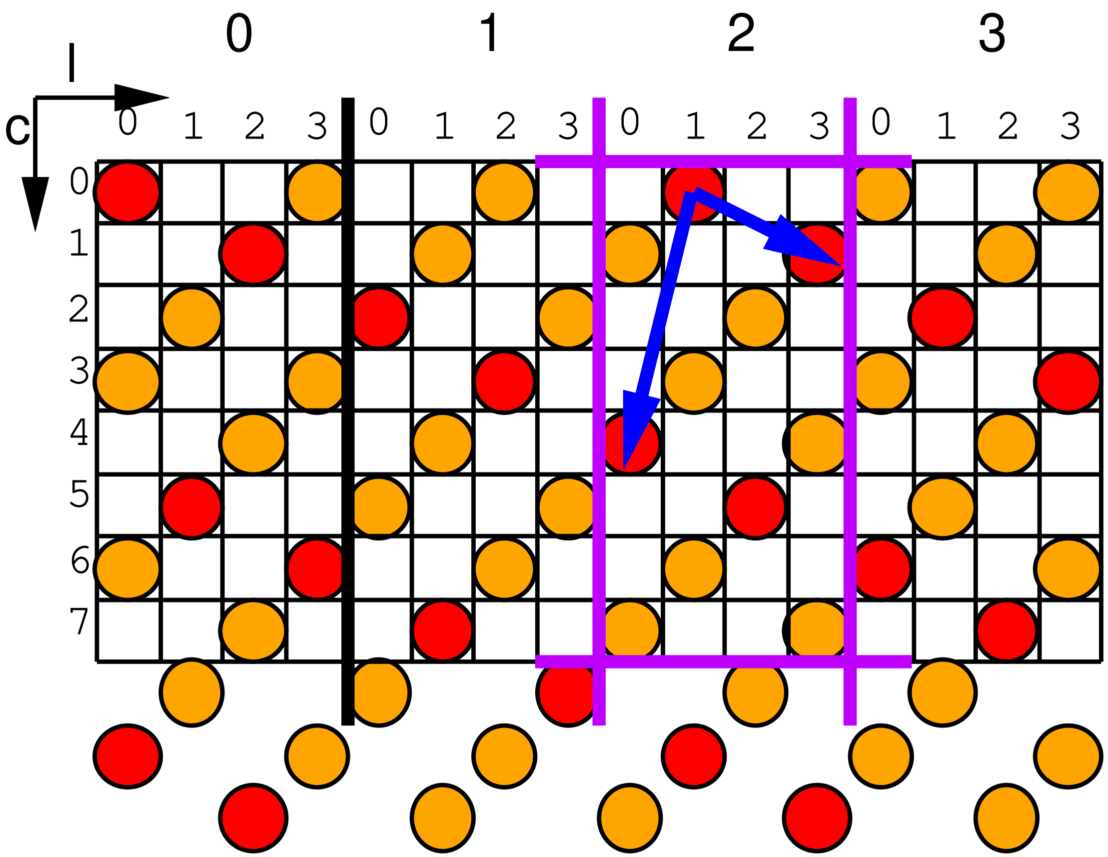
Accès `A(0:42:3)`



Parcours par automate



Approche Z-module et polyèdre



Mise en équation

- variables des objets HPF

  $\texttt{A}(a)$, $\texttt{T}(t)$, $\texttt{P}(p)$

- itérations i

- numéro de bloc l

- numéro de cycle c

Déclarations

- `A(0:42)` : $0 \leq a \leq 42$
- `T(0:127)` : $0 \leq t \leq 127$
- `P(0:3)` : $0 \leq p \leq 3$

Placement

- Align i with 4*i : $t = 4a$
- Dist cyclic(4) : $t = 16c + 4p + l,\ \ 0 \leq l \leq 3$

Itérations

- `A(0:42:3)` : $a = 3i,\ \ 0 \leq i \leq 14$

Linéarité des équations

- équations linéaires $\Rightarrow$ Z-module
- inéquations linéaires $\Rightarrow$ polyèdre
- cadre mathématique simple : algèbre linéaire
- intègre le cas multidimensionnel
- reste simplement à résoudre !

Problèmes à résoudre

- énumération des itérations locales
- caractérisation des données à communiquer
- allocation mémoire des tableaux
- allocation des temporaires
- codes de communication
- codes de calculs

Techniques proposées

pour le cas général

- automate d'état fini (FSM)
  - *Chatterjee et al.* (RIACS, NASA)
  - *Kennedy et al.* (RICE U.)
- résolution *ad hoc*
  - *Paalvast et al.* (Delft)
  - *Gupta et al.* (Ohio)
  - *Stichnoth et al.* (CMU)
  - *Zima et al.* (Vienna)

- cadre linéaire
  - *Hermite, Smith, Fourier...* (XIX siècle)
  - *Ancourt et al.* (CRI, ENSMP)
  - *André et al.* (IRISA)
  - *Amarasinghe et al.* (Stanford)

Remarques

- lien énumération-allocation (caches)
- sections de tableaux...
- peu d'implementations...

Prototypes de compilateur HPF

Ronan Keryell à Supercomputing'93

- outils de recherche

  Fortran D (Rice), VFCS (Vienna), hpfc (CRI)

- outils commerciaux et *freeware*

  Adaptor (GMD), xhpf (APR),

  pour ferme d'AXP (DEC),

  Portland group, VAST-HPF (Pacific Sierra)

Pas beaucoup de vrai HPF !

DEC pour ferme d'AXP

- compilateur difficile à faire, marges faibles
- équipe de 6 personnes
- pas de code intermédiaire généré

Portland group

- associé à Syracuse U., équipe de 4 personnes
- pas de passage de paramètres

  `cyclic` et pas $\neq 1 \rightarrow$ résolution dynamique
- génération de PVM 3

Fortran D (Rice)

– non HPF

– informations statiques et parallélisation

– distributions par block sur une dimension

– vectorisation des messages

– pas d'entrées sorties, pas de résolution dynamique

Vienna Fortran Compiling System

– non HPF, très général $\Rightarrow$ compilation difficile

– interprocédural, interactif

xHPF (Applied Parallel Research)

- non HPF, basé sur Forge 90
- équipe de 5 personnes
- pas de `processors`, une dimention distribuée
- allocation non économe
- extensions `on`, `shrunkcyclic`

VAST-HPF (Pacific Sierra Research)

- simple annonce

Adaptor (GMD)

– non HPF

– directives de CM-Fortran et HPF

– produit distribué gratuitement !

– outil pragmatique et efficace

`hpfc` (CRI)

- Fortran 77 + `independent` + directives statiques
- $\rightarrow$ Fortran 77 + PVM 3
- résolution dynamique
- analyse des recouvrements
- vectorisation des messages
- réductions en parallèles
- Entrées/Sorties

HPF aujourd'hui

- un langage
- quelques compilateurs limités. . .
- types d'applications
- de la recherche
- utilisateurs sceptiques

Le calcul scientifique aujourd'hui

- TMC, KSR, Cray R, Archipel, ACRI, ...
- stations et *clusters* de stations
- petit vectoriel Cray
- machines à mémoire partagée
- utilisateurs (acheteurs) sceptiques

Conclusion

- machines MIMD-DM extensibles
- langage HPF prometteur

  Fortran, standard, data-parallèle

  large classe d'applications concernées
- compilation difficile : taille, sémantique
- efficace si optimisé...
- recherche très active

Pour plus d'informations

– Koelbel, Loveman, Schreiber, Steele, Zosel

The High Performance Fortran Handbook

MIT Press, 1994

– HPF Forum

High Performance Fortran Language Specification

Version 1.1, Nov. 1994

- Forum sur le Web :

  `http://www.erc.msstate.edu/hpff/home.html`

- Tutoriel HPF, Tom Haupt :

  `http://www.npac.syr.edu/hpfa/...`

- Adaptor 2 :

  `http://www.gmd.de/SCAI/lab/adaptor/...`