Langages et compilateurs parallèles

Guy-René Perrin

ICPS
Université Louis Pasteur, Strasbourg

❑ Constats

Remise en cause fondamentale de la programmation

```
{x=0}
    [ x=x+1 ‖ x=2*x ]
{x=1 ou 2}
```

Soumission aux caractéristiques de l'architecture

Evolution de l'architecture des processeurs
et des machines parallèles

❑ Problèmes

L'évolution matérielle est rapide

L'évolution logicielle ne suit pas

Collision des échelles de temps

- Processeurs et architectures :   quelques mois
- Développement des applications :   quelques années
- Exploitation des codes :   quelques dizaines d'années

❑ Objectifs

Modes et outils de programmation standardisés

Portabilité et performances

Sûreté de programmation

Algorithmique et conception des programmes

❑ Moyens

Langages spécifiques et compilateurs

Compilateurs paralléliseurs

Bibliothèques

Environnements de programmation

Langages de programmation parallèle

- Les premiers modes de programmation parallèle

Le temps des désillusions ...

```
tâches, synchronisations,
sections critiques, interblocage, etc.
```

- Les extensions "sournoises" de FORTRAN
- La mise en évidence des difficultés intrinsèques de la programmation parallèle

Programmation bas niveau et non portable

```
REAL SUM,A()
INTEGER I,N,NB_TASKS
COMMON A,SUM,N,NB_TASKS
EXTERNAL WORKER
SUM=0.0
DO I=1,NB_TASKS
  CALL TSKSTART(...,TASK)
ENDDO
DO I=1,NB_TASKS
  CALL TSKWAIT(...)
ENDDO
END

SUBROUTINE TASK
REAL SUML
INTEGER J,N,NB_TASKS
COMMON A,SUM,N,NB_TASKS
SUML=0.0
DO J=ME,N,NB_TASKS
  SUML=SUML+A(J)
ENDDO
CALL LOCKON(SUM)
SUM=SUM+SUML
CALL LOCKOFF(SUM)
RETURN
END
```

Programmation multi-tâches avec variables partagées (Exemple du CRAY X-MP).

```
REAL SUM,TMP,A()
INTEGER I,J,N,NB_NODES,ME
NB_NODES=NUMNODES()
ME=MYNODE()
IF (ME.EQ.0) THEN
  CALL CSEND(N,ALLNDS,...)
ELSE
  CALL CRECV(N,...)
ENDIF
SUM=0.0
DO J=ME+1,N,NB_NODES
  SUM=SUM+A(J)
ENDDO
IF (ME.NE.0) THEN
  CALL CSEND(SUM,0,...)
ELSE
  DO I=1,NB_NODES-1
    CALL CRECV(TMP,...)
    SUM=SUM+TMP
  ENDDO
ENDIF
END
```

Programmation SPMD avec échange de messages (Exemple Intel iPSC).

❏ Constat

Confusion entre mode de programmation
et mode d'exécution

Parallélisme explicite

`processus, espace mémoire, synchronisations`

❏ Etat de la recherche

Les CSP de Hoare

- ❏ `Processus communicants`
- ❏ `Rendez-vous` (cf. OCCAM)
- ❏ Etudes fondamentales : dérivation et preuves de programmes

❑ La programmation des machines SIMD

Un nouveau mode de programmation

- ❑ Séquence d'opérations sur des variables "parallèles"

MPL sur MasPar
C* sur Connection Machine

Parallélisme implicite

- ❑ Définition de variables "parallèles" conformes à une grille de "processeurs virtuels"

```
shape [n][n] matrice;
  float : matrice A, B;
```

- ❑ Opérations globales
- ❑ Expression implicite des communications
- ❑ Diffusions et réductions

❑ Conclusions

L'époque de la confusion des modèles

- ❑ Difficultés de programmation
- ❑ Relative facilité de compilation
- ❑ Absence de portabilité

Critères architecturaux plutôt qu'algorithmiques

- ❑ Granularité des tâches
- ❑ Degré de parallélisme
- ❑ Localité des données

- Enfin Malherbe vint ...

    Virtualisation

    Bibliothèques scientifiques portables

    Bibliothèques de communications

    Une véritable approche de la programmation parallèle

    - Parallélisme de contrôle (passage de messages)
    - Parallélisme de données

❑ Bibliothèques de communications

Approche "moderne" du parallélisme de contrôle

- Abstraction des caractéristiques de l'architecture
- Portabilité et performance des codes

Principes généraux

- Graphe de tâches (programmation SPMD)
- Passage de messages

Produits standards de facto (bibliothèques parfois natives)

- PVM (J. Dongarra et al.)
- MPI   (Forum MPI)

❑ Conclusions

Outils actuellement incontournables

Compromis accepté entre

- ❑ la difficulté de programmation
- ❑ la portabilité des programmes
- ❑ leur performance

Intégration à un mode de programmation
à parallélisme de données avec HPF

❏ Le parallélisme de données

Référence historique : mode d'exécution SIMD

Parallélisme implicite

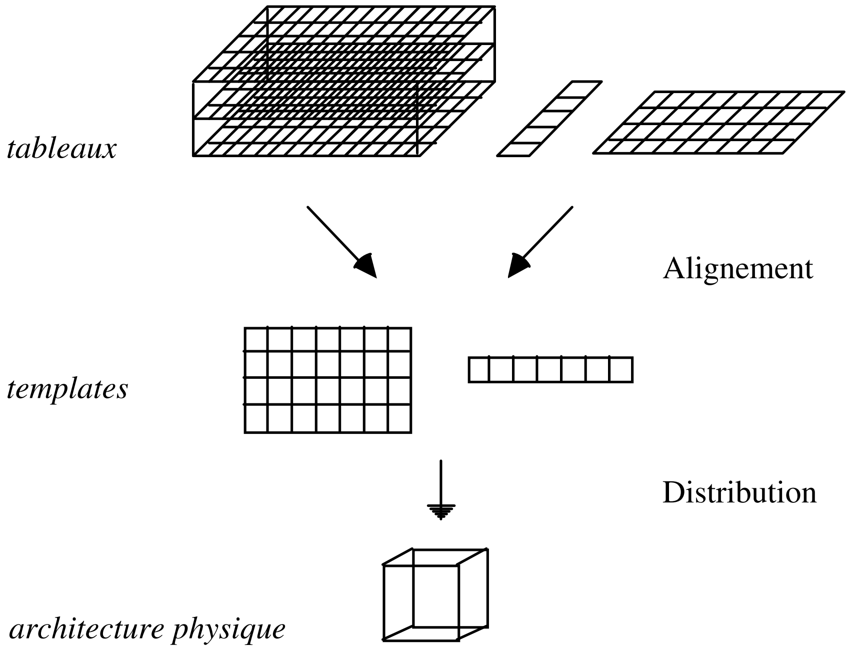
- ❏ Variables "parallèles" conformes à une grille de "processeurs virtuels"
- ❏ Opérations globales et "conditionnement"
- ❏ Expression implicite des communications
- ❏ Diffusions et réductions

+ Alignement des données

- ❏ Directives de compilation `LAYOUT` en CM-Fortran

Un véritable paradigme de programmation parallèle

❑ HPF - High Performance Fortran



- Analyse

  Description "logique" par le programmeur

  - Distribution des données : directives de compilation
  - Parallélisme :

    `opérations globales et instruction FORALL`

    `directive runtime INDEPENDENT`

  Sûreté de programmation

  Objet d'études "académiques"

Compilation

- Des études académiques aux compilateurs du commerce

```
ADAPTOR          Portland Group, Inc
HPFC             Digital Eq. Corp.
```

- Distribution des données et des calculs

  Règle des écritures locales

  Résolution à l'exécution

  Résolution à la compilation :

  — Hypothèses de "régularité"

  — Parcours de polyèdres

Compilateurs paralléliseurs

- Extraction du parallélisme dans un programme
  - Approche de type "transformation de programme"
  - Application "mécanique" de règles prouvées
- Intérêts
  - Réutilisation des codes existants
  - Portabilité

❑ Fondations : l'analyse des dépendances

Dépendance de flot :	`x= ...;` `...= x;`	liée à l'algorithme
Dépendance de sortie :	`x= ...;` `x= ...;`	peut être éliminée
Anti-dépendance :	`...= x;` `x= ...;`	peut être éliminée

❑ Cas intéressants

Eléments de tableaux et boucles imbriquées

Hypothèse : indices de tableaux et bornes de boucles sont des fonctions affines des compte-tours

❑ Transformations "parallélisantes"

Promotion de scalaires en tableaux

Restructuration des boucles

```
éclatement, inversion, etc.
```

Détection de réductions (opérateurs associatifs)

etc.

❑ Outils commerciaux et de laboratoire

Analyse des flots de données et de contrôle

Transformations et restructurations de boucles

```
FORGE,     Applied Parallel Research, Inc
FORESYS,   Simulog
KAP,       Kuck & Associates, Inc

PFC,       Rice
Parafrase,Urbana Champaign
Superb,    Vienna
SUIF,        Stanford
PIPS,        Ecole des Mines de Paris
PAF,       PRiSM, Versailles

etc.
```

❑ Le modèle polyèdrique

Hypothèse : indices de tableaux et bornes de boucles sont des fonctions affines des compte-tours

Les régions de tableaux référencées et le domaine d'itération sont les points entiers de polyèdres

Les transformations sont des changements de base

- ❑ Ordonnancement

  `validité, optimalité`

- ❑ Placement

  `minimisation des communications`

- Actuellement : outils de laboratoire

- Les limites actuelles

  Programmes à contrôle statique

  Difficultés

  - Optimisations : localité des données, préchargement des caches, etc.
  - Analyse interprocédurale
  - Dépendances de contrôle (spéculation)
  - Alias et structures de données dynamiques

  Transformations non linéaires

Quelques conclusions

❑ Qu'offre-t-on à l'utilisateur?

Problèmes à résoudre

- Distribution des données, des calculs
- Définition des communications

Par l'utilisateur	Parallélisme explicite et Passage de messages
Par l'environnement	Parallélisme de données
❑ A la compilation ❑ A l'exécution	Parallélisation automatique

- A quand la crise du logiciel en parallélisme?

Les leçons de l'histoire

- Phase de programmation "bas niveau"

  ```
  Environnements pauvres, non portables,
  Recherche d'efficacité
  ```

- Phase de programmation "haut niveau" (abstraction)

  ```
  Progrès des performances matérielles
  Progrès des compilateurs
  Recherche de sûreté et de portabilité
  ```

- Entre ces deux phases : la crise du logiciel ...

Pour en sortir ...

- Travaux fondamentaux sur les langages, les compilateurs et environnements de programmation
- Nouveaux formalismes "déclaratifs"
- Champ d'investigation largement interdisciplinaire