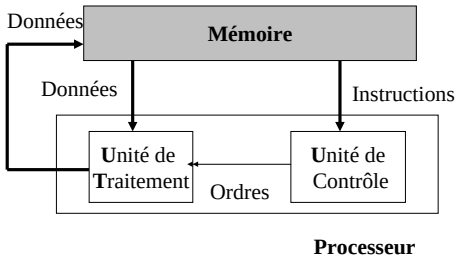
Architectures Parallèles

N. Hameurlain
http://www.univ-pau.fr/~hameur

Plan

- Architecture Séquentielle
- Architecture Parallèle
  - Motivation
  - Modèles
- Multiprocesseurs
- Multicalculateurs

Architecture séquentielle: "de von Neuman"



Architectures parallèles: Motivation

1. Les besoins des applications en puissance de traitement;

2. Les limites de l'approche microprocesseur;

3. L'existence de la propriété du parallélisme dans les applications.

1. Les besoins des applications en puissance de traitement

- La latence du traitement: temps nécessaire pour l'exécution d'un traitement;
- le débit du traitement : nombre de traitement exécutable par unité de temps.

2. Les limites de l'approche microprocesseur(1)

- Les machines séquentielles (un seul processeur) sont construites autour des microprocesseurs (standardisés).
- L'inadéquation du format de données, et des opérations des microprocesseurs aux caractéristiques de certaines applications (traitement d'images, analyse numériques, ...)

Les limites de l'approche microprocesseur(2)

- les limites de l'exécution séquentielle découlent donc des limites des microprocesseurs:
  - capacités d'accès à la mémoire,
  - performance,
  - tolérance aux pannes, etc...

2.1 La limite de capacités d'accès à la mémoire (résolution)

- Multiplier les bancs mémoire;
- Rajouter des mécanismes matériels d'extension d'adressage.

2.2 La limite de performance

- Ne peut être résolue par un microprocesseur même si l'évolution des performances des microprocesseurs suit une courbe exponentielle dans le temps depuis 1985.

3. L'existence de la propriété du parallélisme dans les applications

- Parallélisme de données:
  - la même opération est réalisée par plusieurs processeurs sur des données différentes: exemple : addition de deux matrices;
- Parallélisme de contrôle:
  - des opérations différentes sont réalisées simultanément.
  - indépendance /dépendance du flot (architecture pipeline: dupliquer les unités de traitements).

Architectures parallèles : organisation

- Construites à partir des ressources qui composent les architectures séquentielles: UT, UC, mémoire, entrée/sortie (disque, réseau, etc);
- Durant l'exécution, toutes les unités échangent des informations à travers une ressource supplémentaire: le réseau de communication interne.

Architectures parallèles : classification (1)

- La machine a t-elle un ou plusieurs flux de données (**S**ingle **D**ata stream ou **M**ultiple **D**ata stream) [Flynn 69];
- La machine a t-elle un ou plusieurs flux d'instructions (**S**ingle **I**nstruction stream ou ou **M**ultiple **I**nstruction stream);

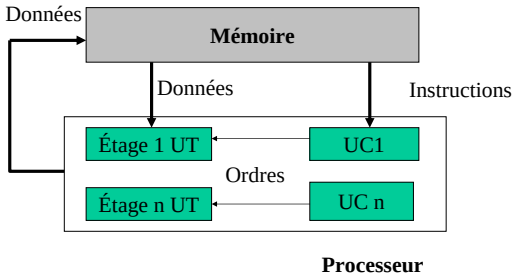
Architectures parallèles : classification (2)

	1 flux d'instructions	>1 flux d'instructions
1 flux de données	Séquentielle	MISD (Pipeline)
> 1 flux de données	SIMD	MIMD

Le Modèle Vectoriel MISD : principe

- Conserve la même structure, mais les UT et les UC sont découpés en étages;
- Un seul flux de données reçoit plusieurs traitement simultanément;
- Il ne s'agit pas de machines multiprocesseurs, mais le parallélisme se situe au niveau plus bas (au sein même du processeur)

Le Modèle Vectoriel MISD: Architecture fonctionnelle



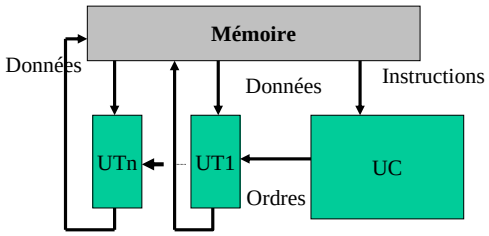
Le Modèle Vectoriel MISD: Exemple

```
x=0;
Tant que la scène est filmée faire
   Ix=F();
   Jx=N(Ix);          // filtrage
   Kx=C(Jx);          //compression d 'image
   S(Kx);             //stocker l'image
   x=x+1;
fin faire
```


Le Modèle SIMD: principe

- Seules les UT sont dupliquées;
- Une seule UC gère le séquencement du programme pour plusieurs UT;
- Toutes les UT fonctionnent en synchronisme et reçoivent les mêmes instructions en même temps;
- La même opération est appliquée à plusieurs données simultanément, puisque chaque UT calcule sur un flux de données différent.

Le Modèle SIMD: Architecture fonctionnelle

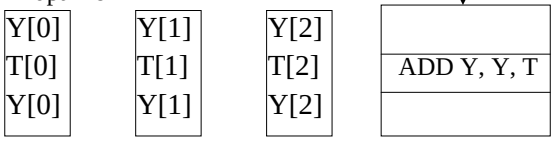


Fonctionnement d'une machine SIMD: Exemple

Parfor i:=0, 2 //for parallèle

Y[i]=Y[i]+T[i] ; → **Y=Y+T**

Endpar for

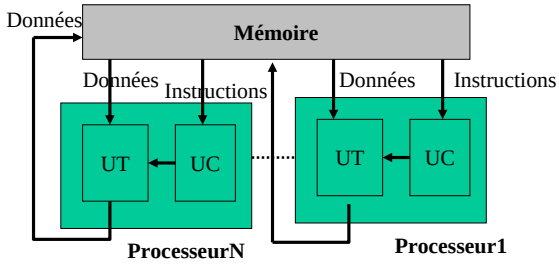


UT1 UT2 UT3 Mémoire du programme

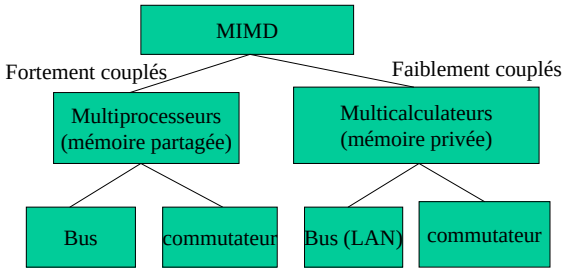
Le Modèle MIMD:principe

- L'architecture du processeur du modèle séquentiel est dupliquée;
- Chaque processeur est libre d'exécuter des instructions qui lui sont propres sur un flux de données qui lui est propre;
- Permet d'exécuter un programme différent sur tous les processeurs;

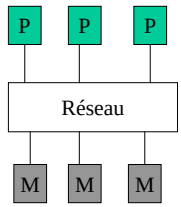
Le Modèle MIMD: Architecture Fonctionnelle



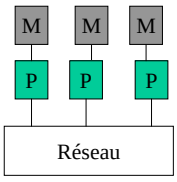
Le Modèle MIMD: classification



Multiprocesseurs/ Multicalculateurs



Mémoire partagée



Mémoire privée

Multiprocesseurs à mémoire partagée : principe

- Les (N) processeurs accèdent à la même mémoire partagée qui doit se comporter comme une mémoire à N ports.
- La mémoire partagée est construite à partir de plusieurs composants mémoire;
- un réseau d'interconnexion relie les composants mémoire et les processeurs;

Multiprocesseurs à mémoire partagée : problèmes

- Un problème de type structurel: la latence des accès mémoire;
- Un problème de type fonctionnel: c'est la coordination des accès à une variable partagée

Multiprocesseurs à Bus : principe

- Un certains nombre d'UC sont connectés à un bus;
- La lecture (ou l'écriture) se fait en mettant l'adresse du mot mémoire sur le bus et en déclenchant le signal approprié (Lecture ou Ecriture);

Multiprocesseurs à Bus : caractéristique

- Simplicité de réalisation;
- Mécanisme de diffusion;
- Accès séquentiel: un seul processeur à la fois peut accéder au bus, et donc à la mémoire.

Multiprocesseurs à Bus : limites et solutions

- Limite: surcharge du bus dès qu'on atteint 4 ou 5 UC;
- Solution: ajouter une mémoire cache entre l'UC et le Bus:
  - le cache conserve les mots mémoire auxquels on a récemment fait accès.
  - tous les accès mémoire passent par le cache;

Cache mémoire des Architectures Multiprocesseurs: principe

- Si le mot mémoire demandé existe dans le cache, le cache répond à l'UC; et aucune demande de bus n'est faite;
- Si le mot mémoire demandé n'existe pas, le bus fait une demande à la mémoire;
- On atteint des taux de succès de plus de 90%, si le cache est assez grand (64 KO à 1 Mo),

Problème de cohérence de cache mémoire

- Deux UC A et B disposent d'un cache;
- Tant que A et B effectuent des références en lecture, ce sont les caches qui renvoient la donnée.
- Si A (par exemple) modifie la donnée:
  - c'est sa copie dans son cache qui sera modifiée.
  - La donnée en mémoire et la copie dans le cache de l'autre processeur ne sont plus cohérentes avec la copie modifiée.

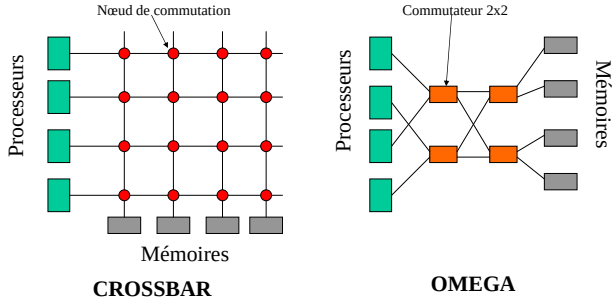
Maintien de Cohérence de cache mémoire

- Technique par *invalidation* des copies blocs:
  - l'écriture sera précédée par l 'invalidation des copies blocs, dans lesquels la donnée réside, présents dans les caches des autres processeurs.
- Technique par *diffusion* des copies blocs:
  - le contrôleur de cache du processeur qui modifie la variable met à jour la mémoire et les copies dans le cache des autres processeurs.

Multiprocesseurs à commutateurs : principe

- Construire 1 Multiprocesseur comportant plus de 64 Processeurs;
- Diviser la mémoire en Modules que l 'on relie aux processeurs (N):
  - CROSSBAR switch: Matrice de commutateurs (NxN noeuds de commutateurs);
  - OMEGA: basé sur les commutateurs 2x2 (Log2(N) commutateurs/étages).

Multiprocesseurs à commutateurs : Exemples



CROSSBAR **OMEGA**

Multiprocesseurs à commutateurs: caractéristiques

- Matrices de commutateurs sont très chères;
- Les grands réseaux OMEGA sont à la fois lents et chers :
  - Si N=1024, il y a 10 étages de commutateurs à franchir;
  - Lecture ou écriture: multiplier par deux le temps de franchir le réseau;

Mémoire partagée : Problème de synchronisation

- Exemple:

```
Do i=1,n
        Parfor j:=2, n-1
        A[j]:=A[j]+A[j-1]+A[j+1]/3
        EndPar for
EndDo
```

- Solution: barrière de synchronisation entre EndPar et EndDo.

Barrière de synchronisation: Réalisation

- A partir d 'une variable partagée initialisée au nombre de processeurs à synchroniser;
- chaque processus qui atteint cette barrière:
  - décrémente cette variable (protégée).
  - se met en *attente active* sur cette variable;
- Lorsque la variable atteint 0, tous les processus se libèrent de l'attente active.

Exclusion Mutuelle : Exemple(1)

Producteur (Prod)	***Consommateur(Cons)***
If compt < N	If compt > 0
Tampon[tete]=item1;	item2=Tampon[queue];
tete=(tete+1) mod N;	queue = (queue+1) mod N;
compt = compt+1;	compt = compt -1
endif	endif

Exclusion Mutuelle (2)

- « compt < N » et « compt > 0 » peuvent être vérifiées simultanément;
- La lecture donne la même valeur par contre seulement l'une des deux écritures en mémoire sera prise en compte:
- La valeur de compt sera incohérente avec l'état du tampon pour le Prod ou le Cons .

Exclusion Mutuelle: réalisation

- Matérielles:
  - Masquage d'interruptions;
  - Instruction indivisible de consultation et de modification d'un mot mémoire: Test and Set (TAS)
- Logicielles:
  - Attente active (Dekker,Peterson..);
  - Sémaphores (Dijkstra), Moniteur (Hoare), ...

Multicalculateurs à mémoire privée: principe

- Chaque UC est directement reliée à sa propre mémoire locale.
- un réseau d'interconnexion gère les échanges entre UC.
- Contrairement aux Multiprocesseurs, le délai de transmission d'un message est long et le débit de transmission est faible.

Multicalculateurs à bus: principe/caractéristique

- La topologie est identique à celle d'un multiprocesseurs à bus;
- Les calculateurs peuvent être des stations de travail, reliées par un réseau Local d'entreprise (LAN);
- Le trafic est moins important que dans les Multiprocesseurs à bus (10 à100 Mbit/s au lieu de 300 Mbit/s et plus).

Multicalculateurs : Exemples

- Treillis:
  - implanté sur des cartes de circuits imprimés;
  - adapté aux problèmes par nature bidimensionnels (théorie des graphes, traitement d ’image)
- Hypercube: cube à n dimensions:
  - chaque sommet représente une UC,
  - chaque arête représente une liaison entre deux UC;
  - chaque UC possède n liaisons avec les autres UC.
  - Adapté à l’implémentation de plusieurs Algo. (Tri, permutation, etc…).