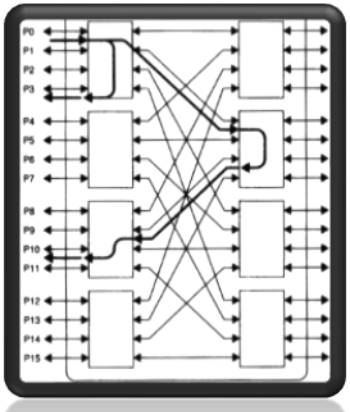
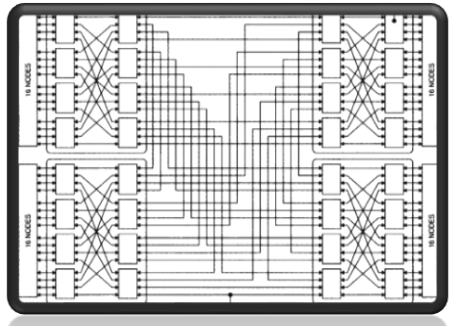


Traitement parallèle de l'information

Dr Maziz Asma

Cours

Master I : Vision Artificielle



Plan

Chapitre 1 :
Introduction : généralités et objectifs

L'architecture des ordinateurs, qu'il s'agisse de microprocesseurs ou de supercalculateurs, est fortement influencée par l'exploitation d'une propriété fondamentale des applications : le parallélisme. Un grand nombre d'architectures présentes dans les sites informatiques sont parallèles. Ce type d'architecture touche une large gamme de machines depuis les PC bi-processeurs jusqu'aux supercalculateurs. Aujourd'hui, la plupart des serveurs sont des machines parallèles (des multiprocesseurs).

Dans ce chapitre nous allons introduire la notion de parallélisme, de discuter la nécessité de l'exploiter pour atteindre de hautes performances et de présenter les principales machines parallèles spécialisées pour le traitement d'images.

1.1. Motivation pour le parallélisme

L'exploitation du parallélisme dans l'architecture des ordinateurs est née de la conjonction de trois éléments :

- Les besoins des applications,
- Les limites des architectures séquentielles
- Et l'existence dans les applications de la propriété de parallélisme.

1.1.1. Besoins des applications

La notion de parallélisme est souvent attachée à celle de la performance d'exécution des applications. Ce dernier terme recouvre différentes notions suivant les besoins des applications. En effet, quel que soit le domaine d'application, le parallélisme peut être exploité pour répondre à deux besoins : la puissance de traitement et/ou la disponibilité.

1.1.2. Limites de l'approche microprocesseur

L'utilisation simultanée de plusieurs ressources entraîne la nécessité de gérer la coordination de ces ressources et complexifie la programmation. Aussi, l'exploitation du parallélisme ne peut se concevoir qu'en réponse aux limitations de l'exécution séquentielle.

Les limites de l'exécution séquentielle découlent donc des limites des microprocesseurs actuels :

- ✓ Performance,
- ✓ Capacité d'accès à la mémoire,
- ✓ Tolérance aux pannes,
- ✓ Variétés des formats de données et des opérateurs.

Le parallélisme permet de palier le problème de la limite de capacité d'accès d'un microprocesseur à la mémoire en multipliant les bancs mémoire et en rajoutant des mécanismes matériels d'extension d'adressage.

1.2. Définitions

1.2.1. Parallélisme :

C'est le parallélisme présent dans une application qui permet d'exécuter simultanément, par des ressources matérielles différentes, plusieurs parties de cette application. Pour introduire la notion de parallélisme nous allons examiner la boucle suivante :

```
Pour i de 1 à n faire
A[i] = B[ i ] + C[ i ]
FinPour
```

Le corps de cette boucle ne comporte qu'une seule opération. Si cette boucle présente du parallélisme, il faut le chercher entre les itérations. Voici les trois itérations :

a) A[1] = B[1] + C[1]

b) A[2] = B[2] + C[2]

c) A[3] = B[3] + C[3]

La sémantique introduite par le programmeur indique les résultants attendus en mémoire après l'exécution de la boucle. Quel que soit l'ordre d'exécution exemple : a, b, c ou c, b, a ou b, c, a etc. les résultats en mémoire sont identiques. La sémantique du programme ne dépend pas de l'ordre d'exécution de cette boucle.

En particulier, l'exécution simultanée de ces trois itérations respecte la sémantique. Les itérations de cette boucle peuvent donc être exécutées en parallèle.

❖ ***Définition formelle***

Pour étudier la présence de parallélisme dans une application, nous avons besoin d'outils plus formels. Bernstein a introduit en 1966 un ensemble de conditions permettant d'établir la possibilité d'exécuter plusieurs programmes (processus) en parallèle.

Supposons deux programmes : Pg1 et Pg2. Supposons que chacun utilise des variables en entrée et produise des résultats en sortie. Nous parlerons des variables d'entrée de Pg1 et Pg2 (respectivement E1 et E2) et des variables de sortie de Pg1 et Pg2 (respectivement S1 et S2).

Selon Bernstein, les programmes Pg1 et Pg2 sont exécutables en parallèle (notation P1 || P2) si les conditions suivantes sont respectées :

$$\{ E1 \cap S2 = \varphi , E2 \cap S1 = \varphi , S2 \cap S1 = \varphi \}$$

Plus généralement, un ensemble de programmes Pg1, Pg2 … Pgk peuvent être exécutés en parallèle ssi les conditions de Bernstein sont satisfaites c'est-à-dire si Pgi || Pgj pour tout couple (i, j) avec $i \neq j$.

L'exemple précédant et les conditions de Bernstein introduisent la notion de dépendances entre deux ou plusieurs programmes (ou opérations). Pour que deux programmes ou deux opérations puissent être exécutées en parallèle, il faut :

1) Qu'ils (elles) soient **indépendants** (elles),
2) Que l'on puisse **détecter cette indépendance** et
3) Qu'il existe suffisamment de **ressources** pour les exécuter simultanément.

1.2.2. Traitement parallèle

C’est l’exploitation d'événements **concurrents** dans le processus de traitement.

- **Concurrence : parallélisme**, **simultanéité** et **pipeline**
  - **Parallélisme :** événements se produisant pendant le même intervalle de temps.
  - **Simultanéité:** évènements se produisant dans le même instant.
  - **Pipeline :** événement se produisant pendant des intervalles de temps chevauchés.

✓ Niveaux du traitement parallèle :

- Travail ou programme (multiprogrammation, temps partagé, multitraitements)
- Tâche ou procédure (décomposition du programme)
- Inter-instruction (analyse de la dépendance des données) ---> vectorisation
- Intra-instruction (au niveau de la micro-programmation) ou du câblage

1.3. Concepts de base

1.3.1. Temps d'exécution

- Temps d'exécution = **T (n)** d'un algorithme parallèle **A** exécuté pour un problème de taille **n** est le temps requis pour exécuter l'algorithme en supposant qu'un nombre illimité de processeurs sont disponibles pour exécuter les diverses opérations.

1.3.2. Coût

- Soit un algorithme **A** utilisé pour résoudre, de façon parallèle, un problème de taille **n** en temps **T (n)**.
- Soit **p(n)** le nombre maximum de processeurs effectivement requis par l'algorithme
- Le **coût** de **A** est alors :

$$C(n) = p(n) \times T(n)$$

1.3.3. Travail

Le travail, **W (n),** dénote le nombre total d'opérations effectuées par l'algorithme parallèle pour un problème de taille **n** sur une machine à **p(n) processeurs**.

1.3.4. Notion d'accélération

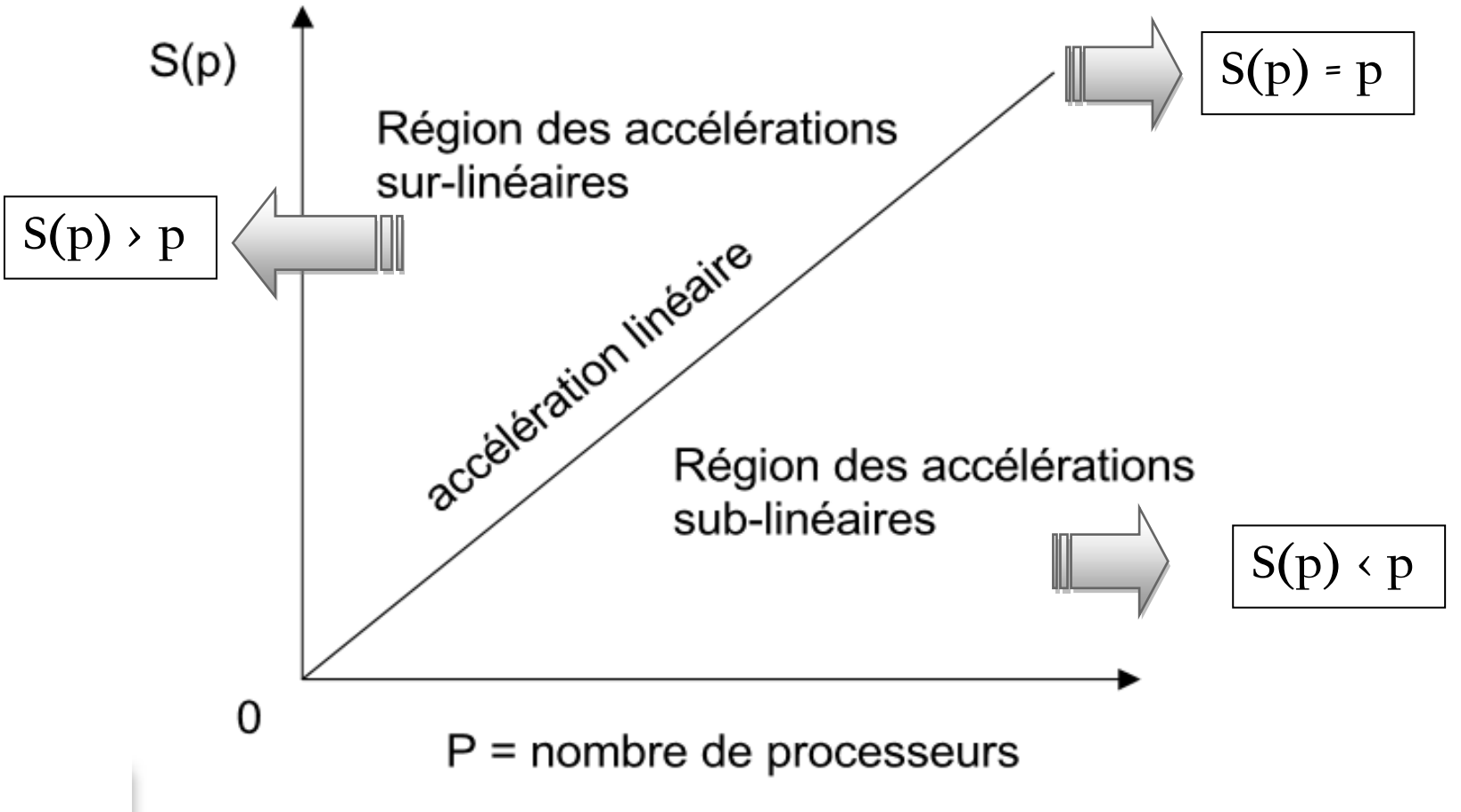
- **Accélération** = **gain de temps** obtenu lors de la parallélisation du programme séquentiel.
- Détermine combien fois un programme parallèle est plus rapide qu'un programme séquentiel.

Définition : Soit :

T1 : le temps nécessaire à un programme pour résoudre le problème **A** sur un **ordinateur séquentiel**,

Tp : le temps nécessaire à un programme pour résoudre le même problème **A** sur un **ordinateur parallèle** contenant **p** processeurs,

Alors l'**accélération** (**Speed-Up**) est le rapport **:** $\mathbf{S(p) = T1 / Tp}$


S(p)
Région des accélérations
sur-linéaires
S(p) = p
S(p) › p
accélération linéaire
Région des accélérations
sub-linéaires
S(p) ‹ p
0
P = nombre de processeurs

Lois d'Amdhal et de Gustafson–Barsis

Notation :

Concept	Notation
Temps partie séquentielle	Ts
Temps partie parallèle	Tp
Accélération	S (p)
Temps pour n taille et p processeurs	T(n, p)

✓ Temps d'exécution

Le temps pour un problème de taille n avec p processeurs peut être décomposé en deux parties :

- **Ts** = partie intrinsèquement séquentielle
- **Tp** = partie pouvant s'exécuter en parallèle

✓ Temps d'exécution pour un processeur : **T (1, n) = Ts + Tp**

✓ Temps d'exécution pour p processeurs : **T (p, n) = Ts + Tp/p**

✓ Accélération : $$S(p) \leq \frac{Ts+Tp}{Ts+Tp/p}$$

a. Loi d'Amdhal

❑ Définition

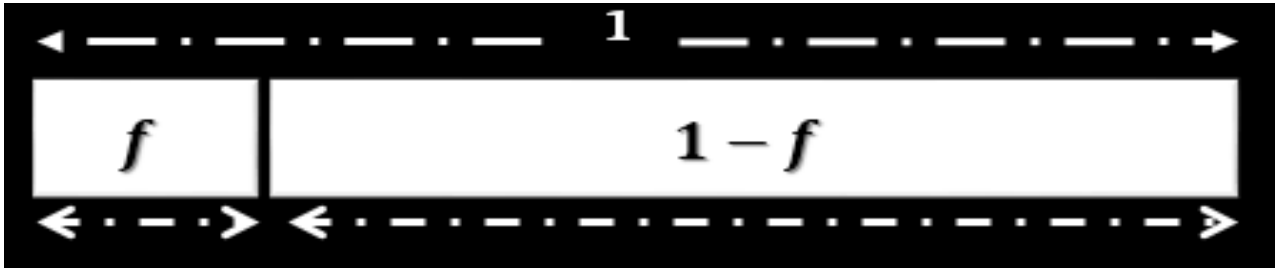
- Soit f la fraction des opérations qui doivent être exécutées de façon séquentielle, avec $0 \leq f \leq 1$.

$$f = \frac{Ts}{Ts+Tp}$$

- Alors, l'accélération maximale **S(p)** pouvant être obtenue avec **p** processeurs est :

  - $\mathbf{S(p)} \leq \frac{1}{f+(1-f)/p}$

- ✓ La loi d'Amdahl ≈ on normalise le temps d'exécution séquentielle (à 1)



- ✓ La formulation de la loi d'Amdahl ignore les sur-coûts associés à l'exécution parallèle, c'est-à-dire, les coûts des synchronisations et communications entre processus.
- ✓ Pour un nombre fixe de processeurs, l'accélération va augmenter lorsqu'on augmente la taille du problème.

b. Loi de Gustafson–Barsis

- La loi d'**Amdhal** suppose que l'objectif est de produire le résultat le plus rapidement possible.
- L'effet **Amdhal** s'explique par le fait que lorsqu'on augmente la taille du problème, la fraction du **Ts** diminue.
- Pour la loi de **Gustafson**–**Barsis**, on va supposer que le temps d'exécution parallèle (**Tp**) est fixe.

❑ Définition

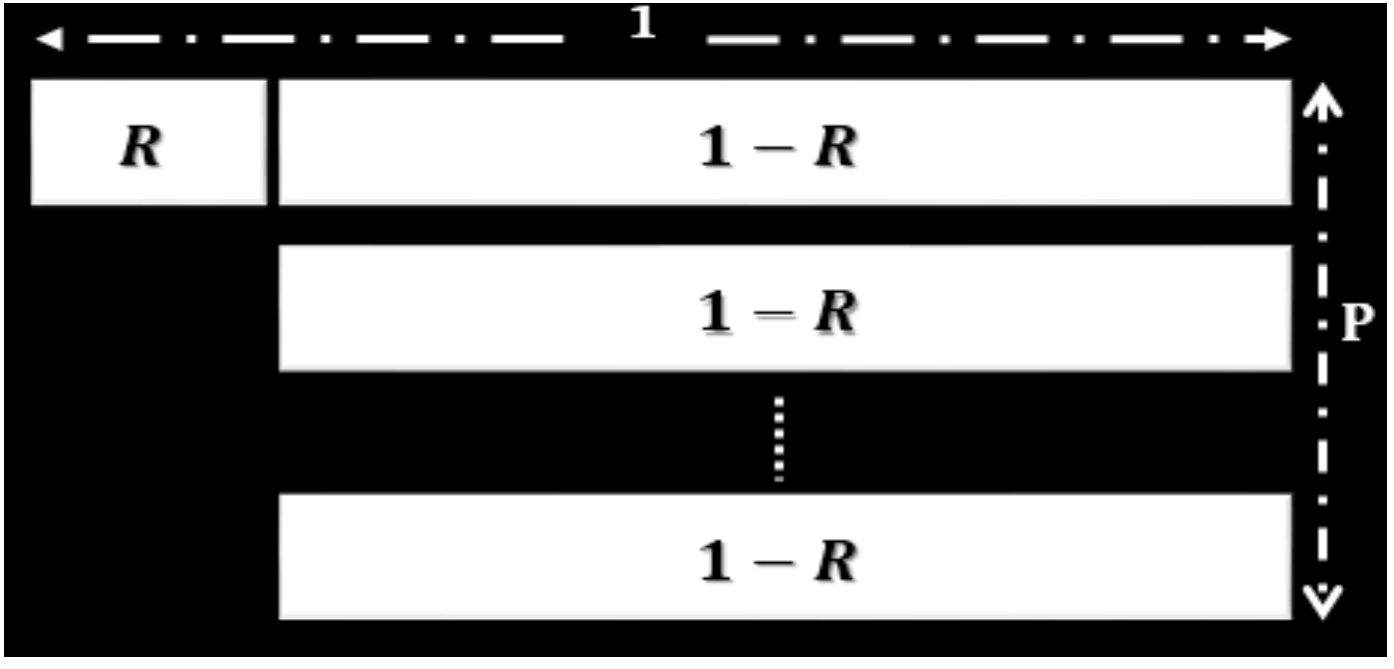
➢ Pour un problème de taille **n** traité avec **p** processeurs, soit **R** la fraction du temps d'exécution (parallèle) consacrée à la partie intrinsèquement séquentielle (avec $0 \leq \mathbf{R} \leq 1$).

$$R = \frac{Ts}{Ts + Tp/P}$$

➢ Alors, l'accélération maximale **S(p)** pouvant être obtenue est la suivante :

$$\mathbf{S(p) = p + (1 - p)\ R}$$

✓ Loi de **Gustafson**– **Barsis** ≈ on normalise le temps d'exécution parallèle.



1.3.5. Notion d'efficacité

- L'efficacité nous indique dans quelle mesure les divers processeurs sont bien utilisés ou pas.
- Mesure le taux d'occupation « **utile** » des processeurs.

Définition : Soit

- **$T_1(n)$** le temps nécessaire à l'algorithme pour résoudre une instance de problème de taille ***n*** avec un seul processeur,
- **Tp(n)** celui que la résolution prend avec **p** processeurs,
- **$S(n,p) = T_1(n) / Tp(n)$** le facteur d'accélération.

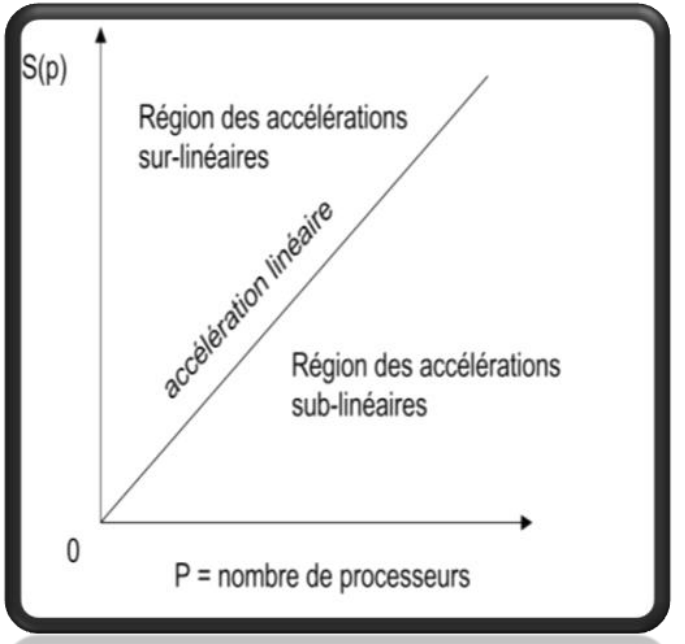
On appelle efficacité de l'algorithme le nombre :

$$E(n,p) = S(n,p) / p$$

- Donc, l'efficacité est la normalisation du facteur d'accélération.
- Généralement exprimé en pourcentage.

Efficacité/accélération

- Une accélération **linéaire** correspond à un gain de temps égal au nombre de processeurs (**100% activité**),
- Une accélération **sub-linéaire** implique un taux d'activité des processeurs < **100 %** (communication, coût du parallélisme...)
- Une accélération **sur-linéaire** implique un taux d'utilisation des processeurs > **à 100 %** ce qui paraît impossible (selon la loi **d 'Amdhal**).



1.3.6. Puissance de calcul

- Calculer en **MIPS ou FLOPS**

  - **MIPS** (**M**achine **I**nstructions **P**er **S**econd) représente le nombre d'instructions effectuées par seconde.

  - **FLOPS** (**FL**oating Point **O**perations **P**er **S**econd) représente le nombre d'opérations en virgule flottante effectuées par seconde.

- Des tailles typiques millions, milliards, trillions ...
  - ✓ **Mega** : Mflop/s = 10^6flop/sec
  - ✓ **Giga :** Gflop/s = 10^9flop/sec
  - ✓ **Tera :** Tflop/s = 10^{12}flop/sec
  - ✓ **Peta :** Pflop/s = 10^{15}flop/sec
  - ✓ **Exa** : Eflop/s = 10^{18}flop/sec
  - ✓ **Zetta :** Zflop/s = 10^{21}flop/sec
  - ✓ **Yotta** : Yflop/s = 10^{24}flop/sec

- Par exemple **PERF 2002** parmi les certains processeurs vectoriels ont une puissance de calcul de 300 Mflops.
- Parmi les machines la plus puissante au monde actuellement 34 Pflop/s (2013) consomme 17.8MW.

❑ Exemples des machines parallèles

a) Centres nationaux

❑ **CINES : C**entre **I**nformatique **N**ational de l'**E**nseignement Supérieur (Montpellier)

- **IBM SP2** (207 **proc**. P2SC et 112 **proc**. Power3).
- **Origin2000** (256 **proc**. R12000).

❑ **IDRIS** : **I**nstitut du **D**éveloppement et des **R**essources en **I**nformatique **S**cientifique (Orsay)

- SGI-Cray T3E (256 proc.)
- Fujitsu VPP300 (8 proc.)

b) Organismes publiques

- **CEA**
  - Grenoble : Compaq SC232 (232 procs EV67 à 667 MHz),
  - Bruyères-Le-Chatel : SGI-Cray T3E (176 proc.), T90 (24 proc.).
- **Météo France, Toulouse**
  - Fujitsu VPP700E (26 proc.)

c) Organismes privés

- **Renault**
  - **Origin2000** (32 proc.)
  - **NEC SX-5** (10 proc.)
  - **Cray SV-1** (32 proc.)

Le top 500 des ordinateurs les plus rapides au monde

http://www.top500.org

année	ordinateur	p	R_{peak}	N	R_{max}
1995	Fujitsu	140	206 Gflop/s	42 000	170 Gflop/s
1997	Intel	9152	1.8 Tflop/s	235 000	1.34 Tflop/s
2000	IBM ASCI White	8192	12 Tflop/s	430 000	4.9 Tflop/s
2001	IBM ASCI White	8192	12 Tflop/s	518 096	7.2 Tflop/s
2002	NEC Earth Simulator	5120	41 Tflop/s	1 075 200	36 Tflop/s
2004	IBM Blue Gene	32'768	91.7 Tflop/s		70.7 Tflop/s
2005	IBM Blue Gene	131'072	367 Tflop/s	1'769'471	280 Tflop/s
2007	IBM Blue Gene	212'992	596 Tflop/s		478 Tflop/s
2008	IBM roadrunner	129'600	1.45 Pflop/s		1.1 Pflop/s

Tel que :

- **P** est le nombre de processeurs,
- **N** est la taille du problème,
- $\mathbf{R}_{\mathbf{preak}}$ est la puissance de crête,
- $\mathbf{R}_{\mathbf{max}}$ est la puissance réalisée sur le problème.

1.4. Les ordinateurs parallèles

Généralement, Il existe trois types d'ordinateurs parallèles :

a) Les machines vectorielles multi-processeurs :

- ✓ Faible nombre de processeurs puissants (1 à 16).
- ✓ Mémoire partagée.
- ✓ Limite atteinte, coût important.

b) Les multi-processeurs à mémoires distribuées :

- ✓ Grand nombre de processeurs ordinaires à mémoire locale.
- ✓ Communication par envoi de messages à travers des réseaux de communication.
- ✓ Les processeurs ont leur propre séquenceur.

c) Les machines synchrones :

- ✓ Très grand nombre d'éléments de calcul (4096 à 65536) de faible puissance avec une toute petite mémoire locale.
- ✓ Un séquenceur unique : exécution d'une même instruction sur des données différentes.

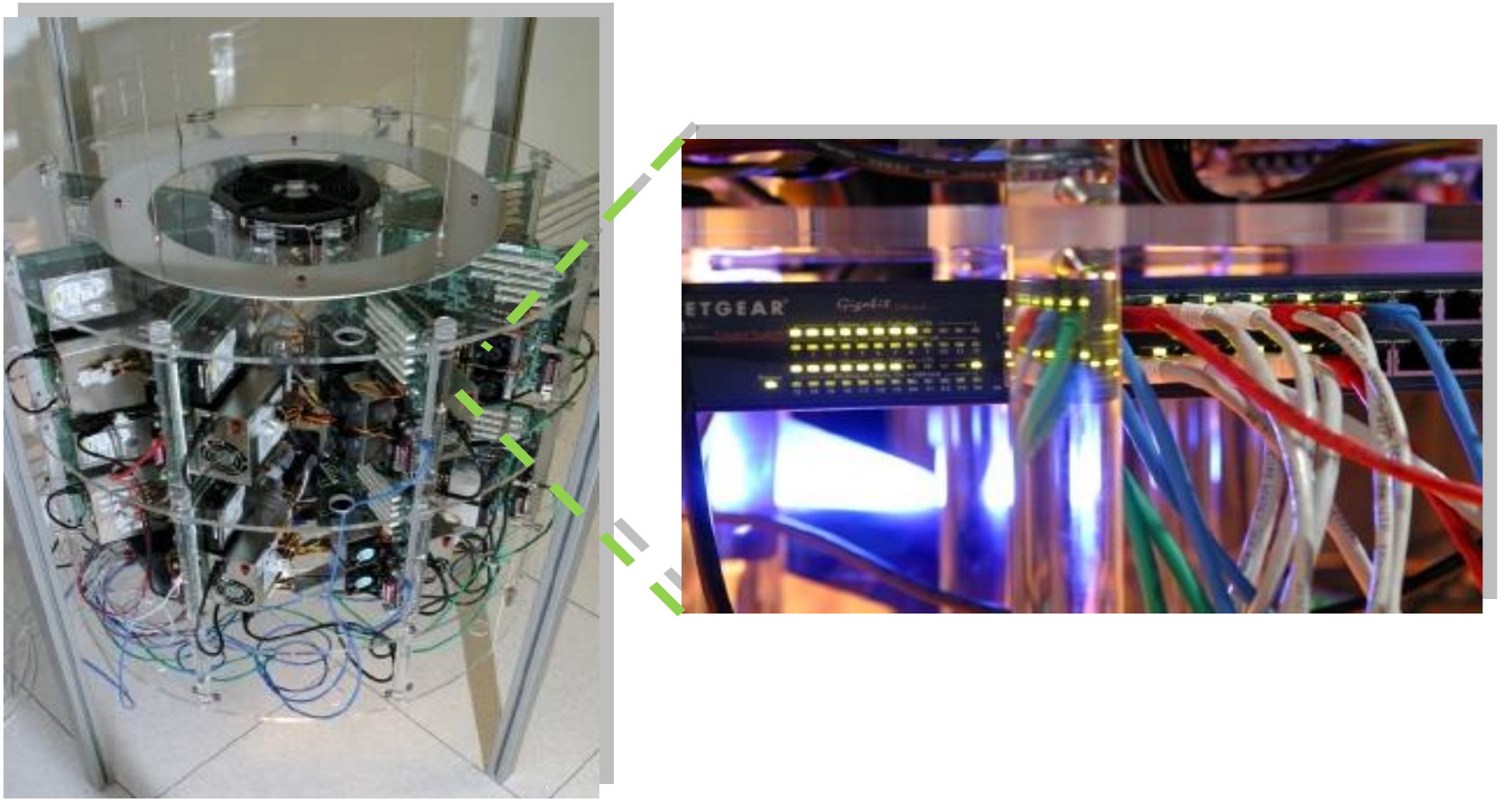
- **Maintenant :** on se dirige vers l'utilisation des **multi-processeurs à mémoires distribuées.**
- **Futur :** on se dirige vers l'utilisation des stations de travail reliées par un réseau à très haut débit.

Exemple :

- ❖ Développement de bibliothèques de communication (**PVM**, **MPI**) Méta-Computing, Grille de calculs…

Exemple : La grappe de PC conçue par hp Laboratories Grenoble et INRIA Rhône-Alpes

Exemple : la grappe de l'INRIA



Super-ordinateurs

- Le earth-simulator au Japon,
- 5000 processeurs…