Architectures parallèles

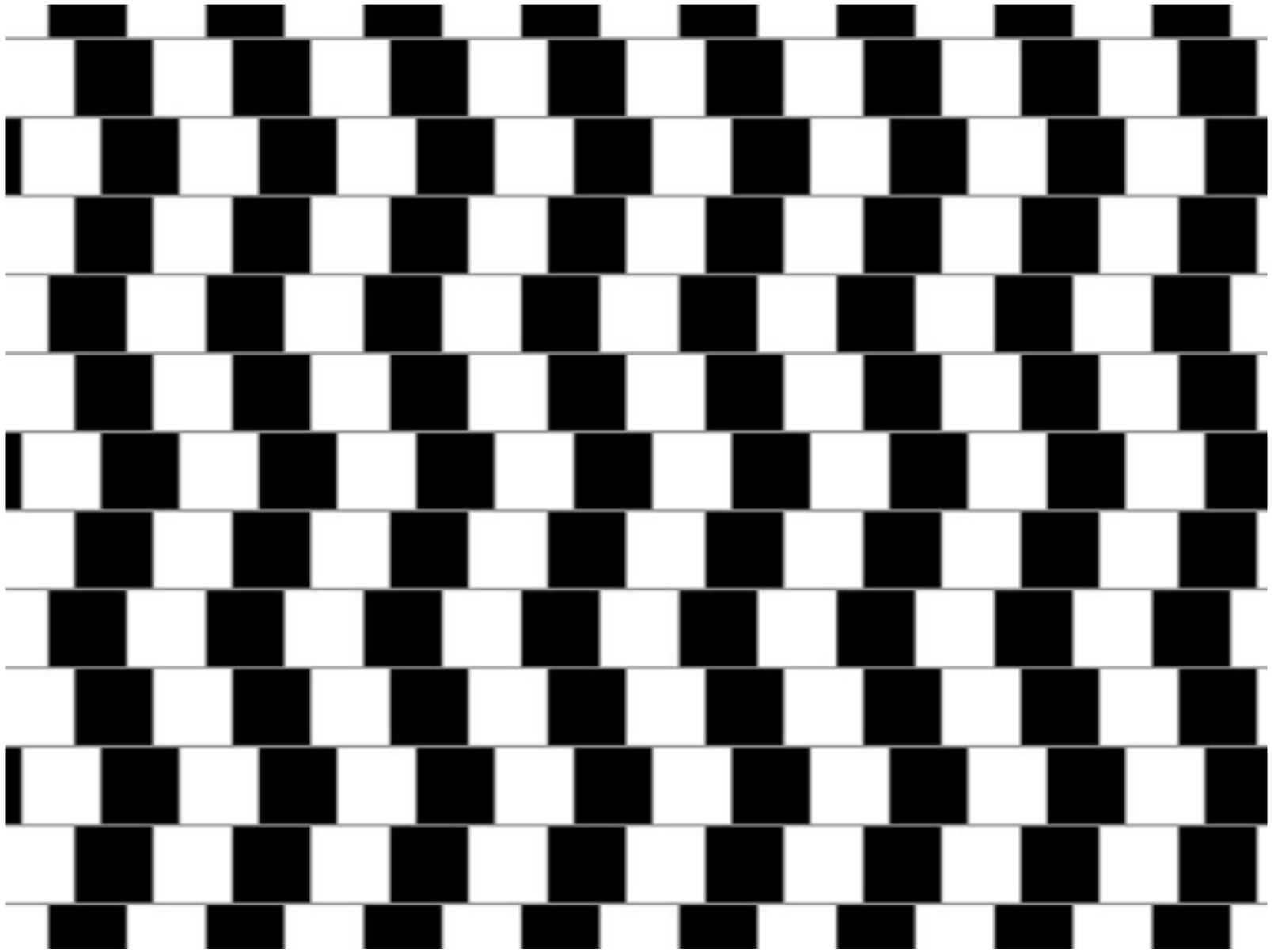


Image: Michael Thompsett

GIF-1001 Ordinateurs: Structure et Applications, Hiver 2015

Jean-François Lalonde

Merci à Étienne Tremblay

Architecture parallèle

- Architecture parallèle: permet l'exécution de deux tâches différentes (ou plus!) simultanément.

- La taxinomie de Flint classifie les architectures parallèles:

  - **SISD**, "Single Instruction Single Data": ne traite qu'une seule donnée par instruction.
    - exemple: vieil ordinateur avec un microprocesseur unique.

  - **SIMD**, "Single Instruction Multiple Data": une instruction est exécutée sur plusieurs données. On retrouve dans cette catégorie les microprocesseurs vectoriels ou matriciels qui traitent effectuent un calcul, en parallèle, sur tous les éléments d'un vecteur ou d'une matrice.
    - Exemple: traitement de signal et cartes graphiques.

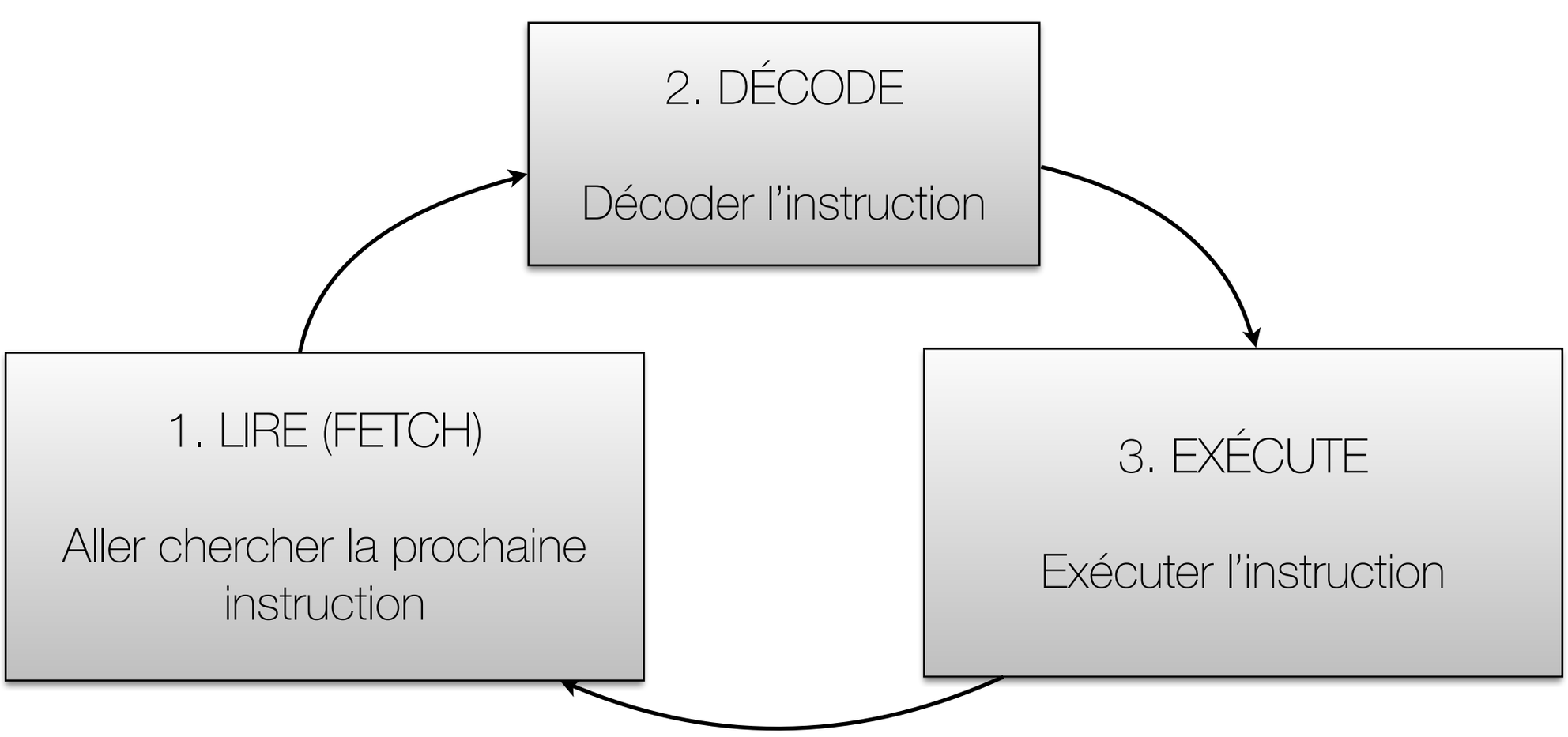
  - **MISD**, "Multiple Instruction Single Data": les instructions sont exécutées plusieurs fois sur une donnée unique. Ce genre de système est quasiment inexistant.
    - Exemple: applications très spécialisées comme l'aérospatial où l'on peut effectuer plusieurs calculs sur la même donnée afin d'assurer la redondance.

  - **MIMD**, "Multiple Instruction Multiple Data": plusieurs processeurs traitent, en parallèle, plusieurs données. Il s'agit du cas le plus commun.
    - Exemple: votre ordinateur personnel.

Cycle d'instructions

Que fait le microprocesseur?

1. Lire: aller chercher la prochaine instruction
2. Décode: décode l'instruction (détermine ce qu'il y a à faire)
3. Exécute: exécuter l'instruction



Pipelines

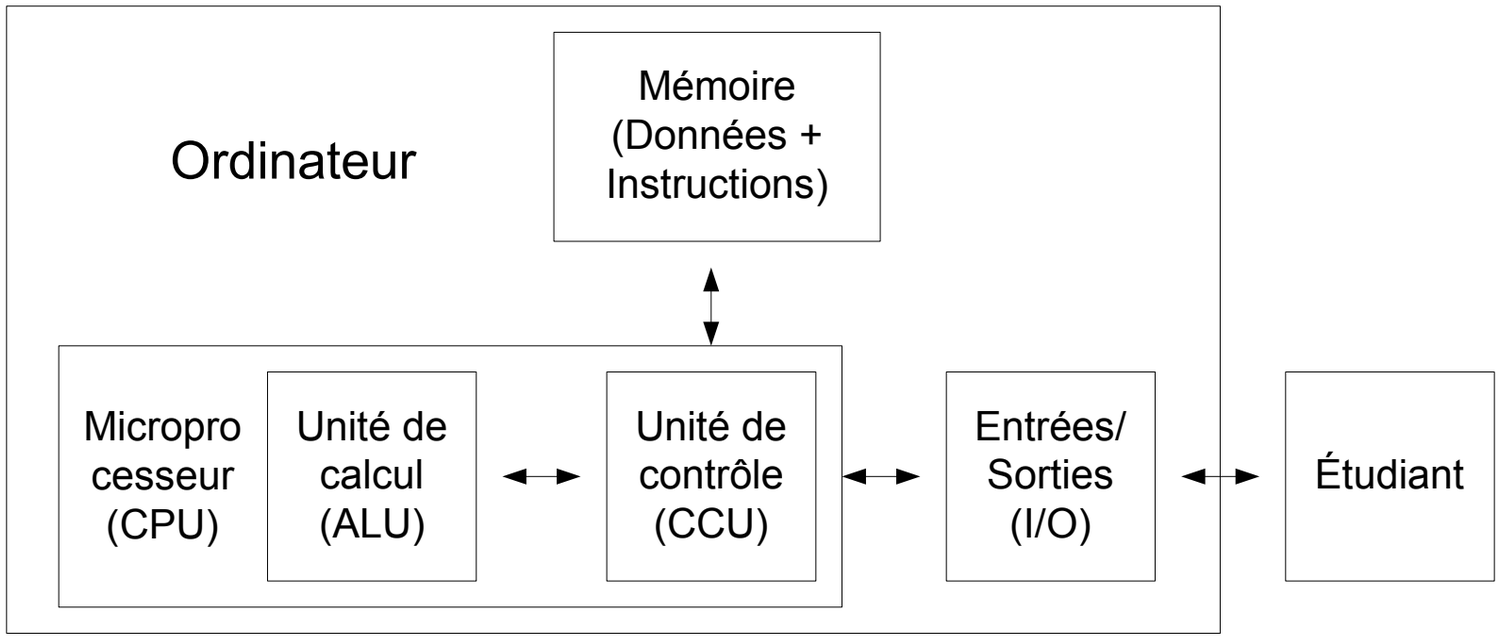
- Début des années 1990
- Idée: séparer l'architecture en deux unités séparées, pouvant être exécutées simultanément
- Comme dans un restaurant:
  - "Fetch-decode": les serveurs vont prendre les commandes des clients aux tables
  - "Execute": les cuisiniers préparent les repas

Pipelines

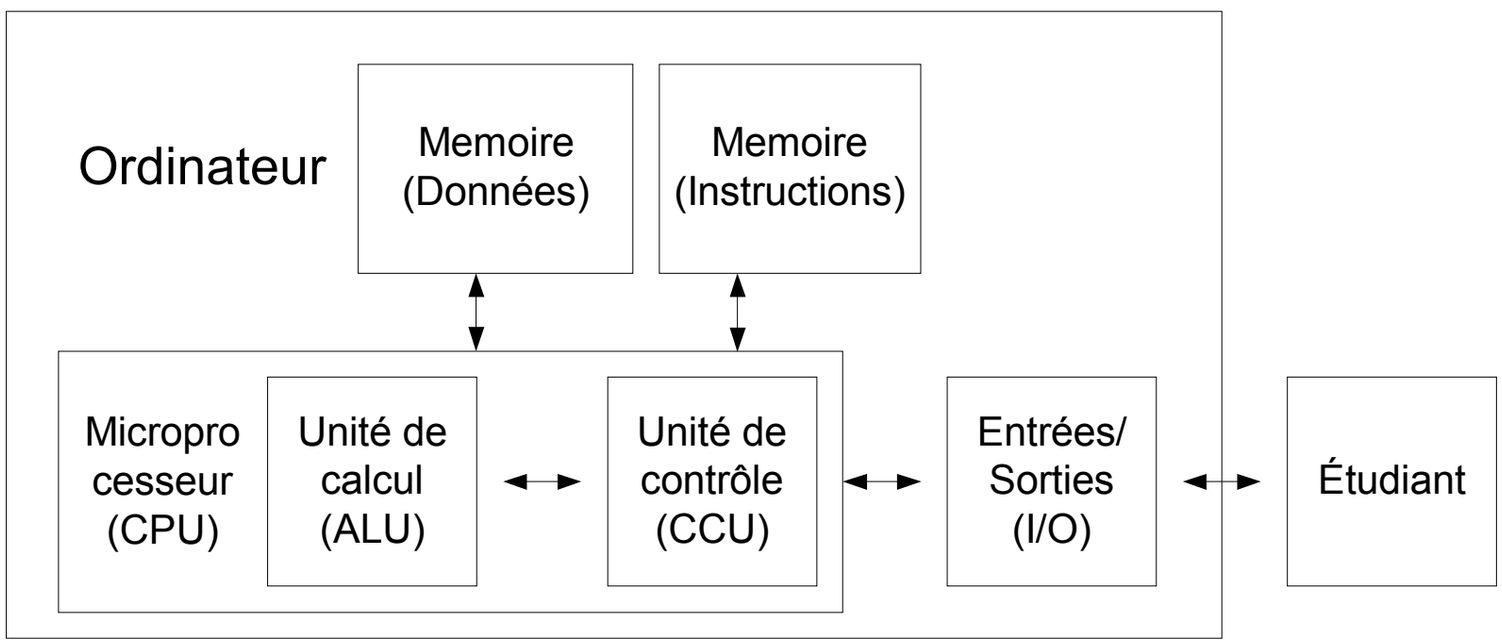
- Qu'est-ce qui limite le temps d'exécution?
  - L'étape la plus longue
    - (exécution, car il faut calculer et lire les opérandes, exécuter l'instruction, écrire le résultat…)
- Solution?
  - Raccourcir l'étape la plus longue, soit diviser la pipeline en plusieurs étages.
  - Exemple:
    - Fetch (lecture) l'instruction
    - Decode l'instruction
    - Calcul des opérandes
    - Fetch (lecture) des opérandes
    - Exécute l'instruction
    - Écrire le résultat

Architecture de Harvard

von Neumann



Harvard



Images: Étienne Tremblay

Pipelines — branchements

- Que faire si l'on rencontre un branchement:
  - Inconditionnel:
    - On continue à l'adresse subséquente!
  - Conditionnel:
    - Il faut vider le pipeline si le branchement doit être pris…
    - Pendant ce temps, les unités de notre pipeline sont inactives!
- Solution?
  - Plusieurs solutions sont possibles, par exemple la prédiction de branchement

Pipeline — dépendances

- Que faire si l'on rencontre une dépendance
  - Les opérandes d'une instruction dépendent du résultat d'une instruction précédente:

```
ADD R1, R0, R0
LDR R0, [R1]
SUBS R3, R2, #1
```

- Solution?
  - On peut exécuter les instructions en désordre!
    - Par exemple: on peut exécuter l'instruction SUBS en attendant que le résultat de ADD soit prêt!

Pipeline d'instructions — notes

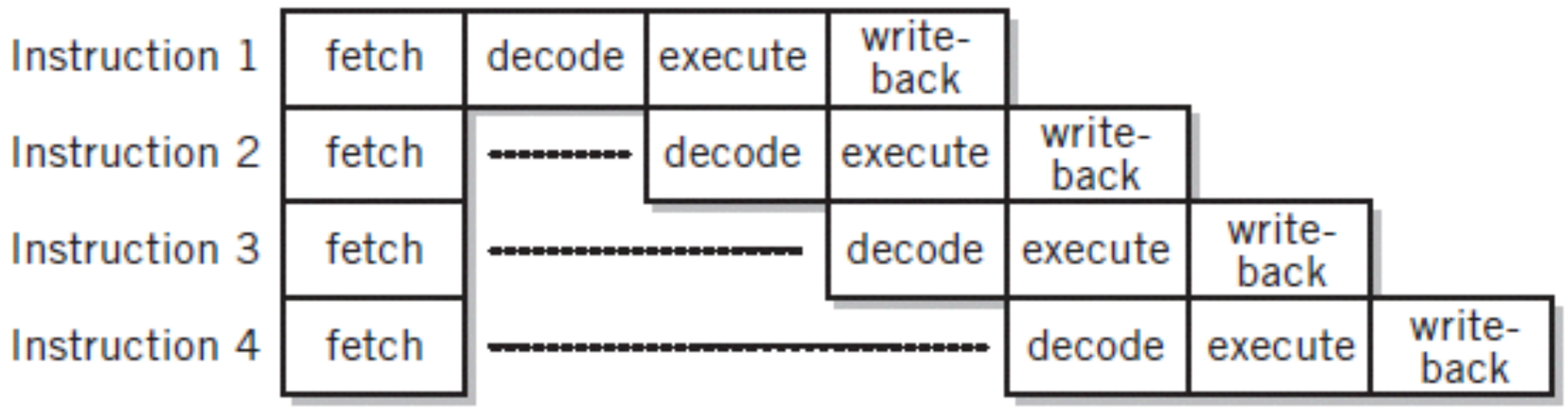
- Dans un pipeline d'instructions, l'exécution de chaque instruction est découpée en parties. Des parties différentes d'instructions différentes sont exécutées simultanément.

- Avantages:
  - Les ressources de l'unité d'exécution sont exploitées au maximum. À chaque coup d'horloge, on peut produire le résultat d'une instruction (dans un monde idéal).
  - Si on découpe l'exécution d'une instruction en étapes très petites, on peut augmenter la fréquence d'horloge.

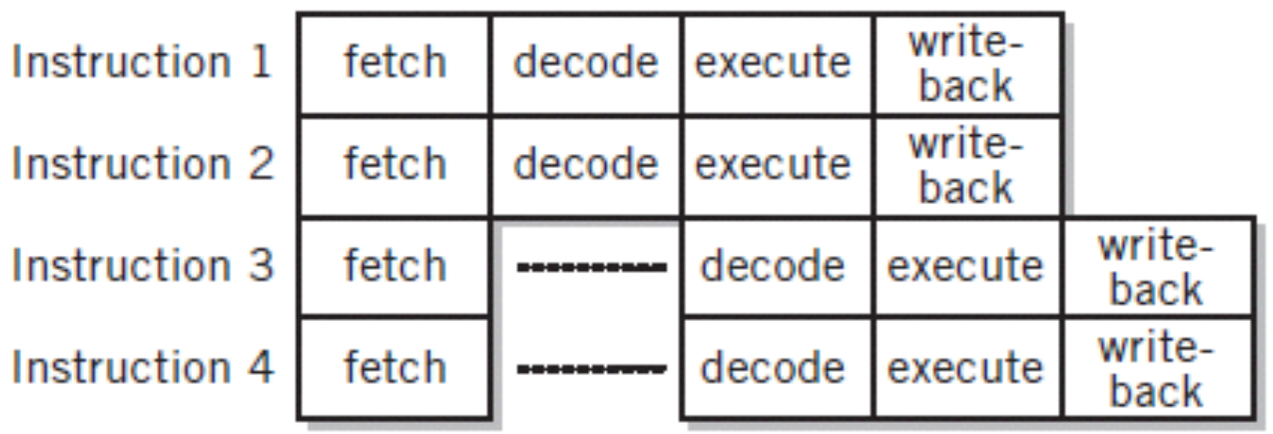
- Limites:
  - Les étapes d'une instruction n'ont pas toutes la même durée. Dans un pipeline, les durées des étapes sont forcément égales et déterminées par l'étape la plus longue.
  - Des dépendances entre les instructions, des branchements, ou l'accès par plusieurs instructions aux mêmes ressources diminuent le gain apporté par les pipelines
  - Il faut du temps, du matériel supplémentaire et des algorithmes plus complexes pour propager les informations d'un étage à l'autre du pipeline.

Architectures “superscalaires”

- Vers 1993: pourquoi se limiter à une seule pipeline?

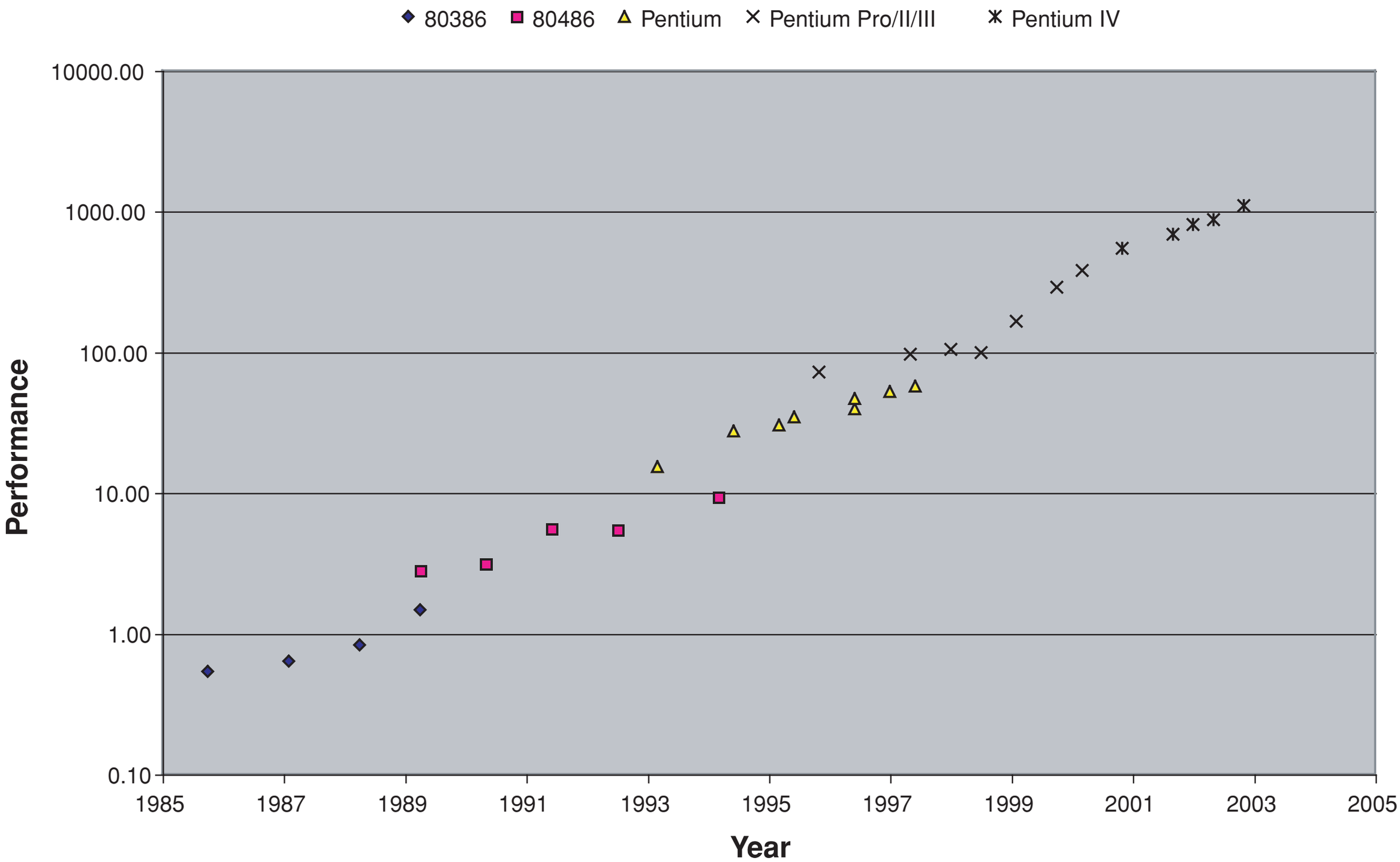


(a) Scalar



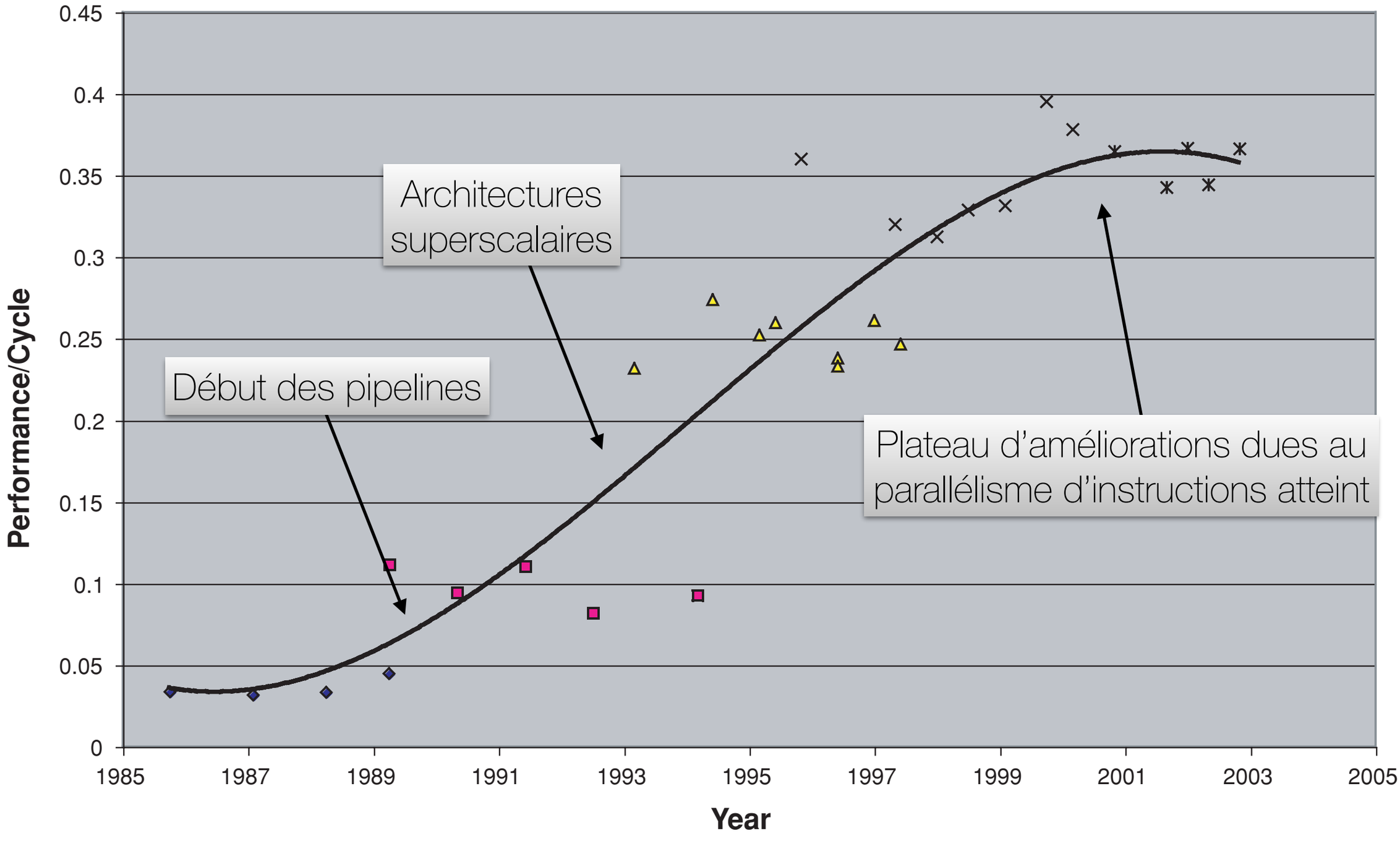
(b) Superscalar

Progression — performance



Source: Olukotun et al., 2007

Progression — performance/# de cycles



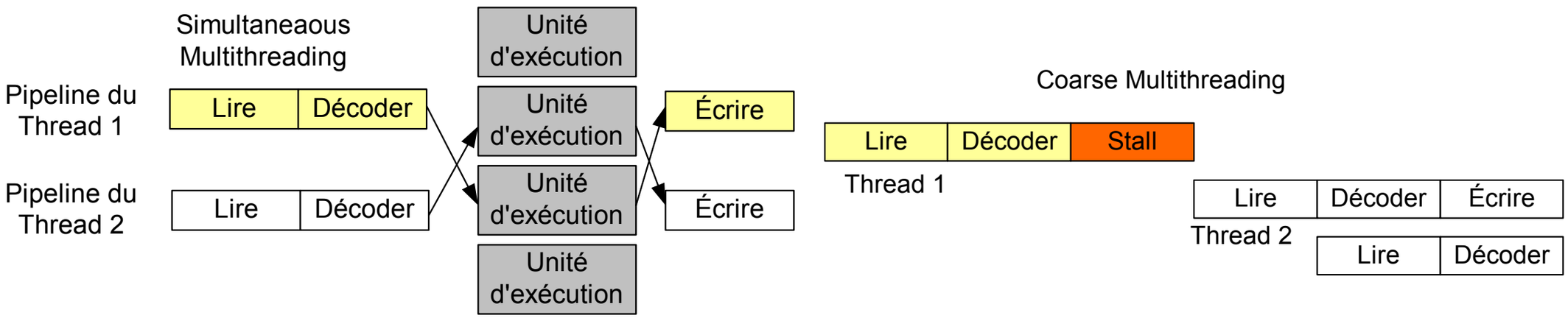
Source: Olukotun et al., 2007

ILP vs TLP

- L'exploitation du ILP (Instruction Level Parallelism) est l'exécution en parallèle des instructions d'un thread unique.
  - pipelines
  - pipelines superscalaires
- L'exploitation du TLP (Thread Level Parallelism) est l'exécution en parallèle des instructions de processus ou threads différents.
  - multithreading
- Habituellement, les instructions d'un thread unique sont exécutées par un unique processeur.

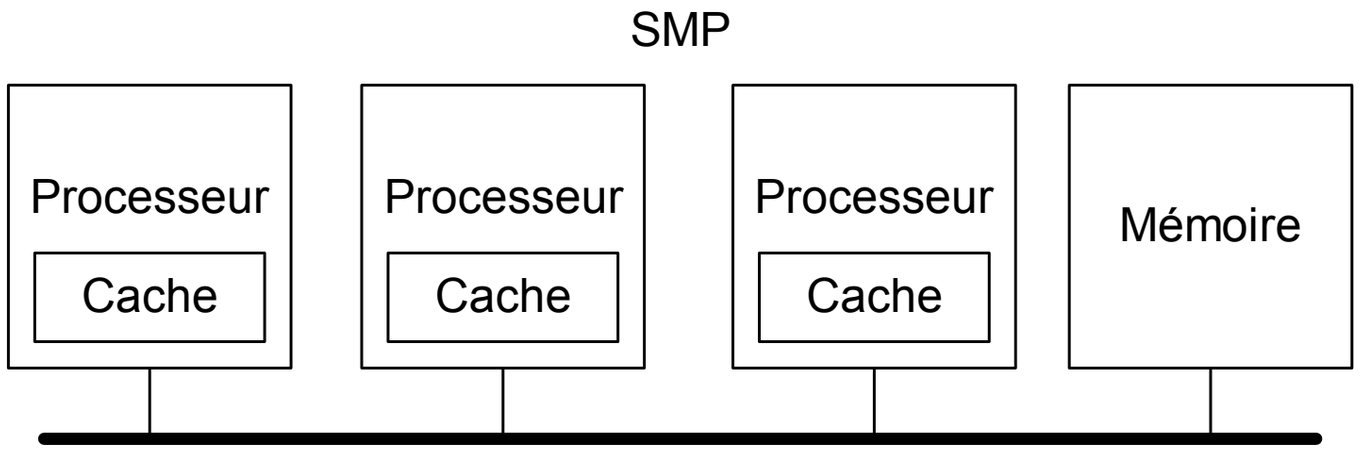
Multi-threading

- Description: partage des ressources d'un processeur par plusieurs processus simultanément
- Avantages: Permet de limiter les pertes de temps lorsqu'un thread est bloqué (lorsque l'instruction désirée n'est pas dans la cache par exemple). Permet une utilisation plus optimale des unités d'exécution. En moyenne, augmente habituellement les performances autour de 20%
- Limites:
  - Il faut une copie du contexte de chaque processus: e.g. fichier des registres, PC
  - Il faut la capacité de changer de processus (thread switch) rapidement.
  - Il faut une copie de certaines ressources du microprocesseur (exemple: lecture simultanée d'instructions de deux thread différents).
  - Il est possible de ne pouvoir exécuter aucun thread.



Multiprocesseurs avec mémoire unique (SMP)

- Description: Plusieurs processeurs partagent une mémoire et y échangent les informations nécessaires à l'exécution de plusieurs thread en parallèle.
- Avantages:
  - Système multiprocesseurs simple souvent utilisé lorsque le nombre de processeur est restreint (disons < 100).
- Limites:
  - La mémoire doit avoir suffisamment de bande passante pour fournir des informations à tous les microprocesseurs.
  - Lorsque des caches sont utilisées afin de diminuer le nombre d'accès à la mémoire (très souvent!), il faut gérer la cohérence des données (voir plus loin).

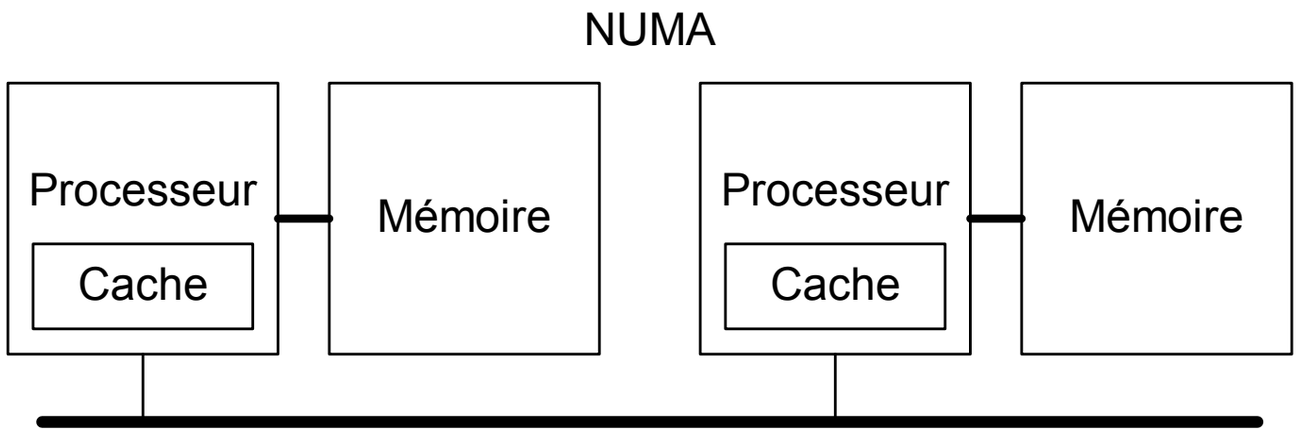


Accès non-uniforme à la mémoire (NUMA)

- Description: Lorsque le nombre de processeurs devient grand (>100), une mémoire unique ne peut répondre aux besoins de tous les microprocesseurs sans que son prix n'explose. La solution consiste à partager la mémoire entre les microprocesseurs. Chaque microprocesseur contrôle une partie de la mémoire.
    - On parle d'accès non-uniforme parce que le temps d'accès à la mémoire dépend de l'adresse. Si un processeur accède à la mémoire qu'il contrôle, le temps d'accès sera plus court que s'il accède à la mémoire d'un autre processeur.
    - Les adresses peuvent être communiquées à tous les processeurs (déterminées lors de l'installation du système), ou chaque processeur peut avoir son système d'adresse (lors d'accès à une mémoire distante, il faut spécifier la mémoire et l'adresse dans la mémoire)
- Avantages: Permet de supporter des systèmes très larges avec des contraintes acceptables sur les accès mémoires.

Accès non-uniforme à la mémoire (NUMA)

- Limites:
  - Lorsque des caches sont utilisées afin de diminuer le nombre d'accès à la mémoire (très souvent!), il faut gérer la cohérence des données.
  - Il faut ajouter du matériel et/ou du logiciel pour assurer la cohérence des données.
  - L'accès à la mémoire est plus complexe
  - Il faut un bus d'interconnexion pour relier les microprocesseurs entre eux

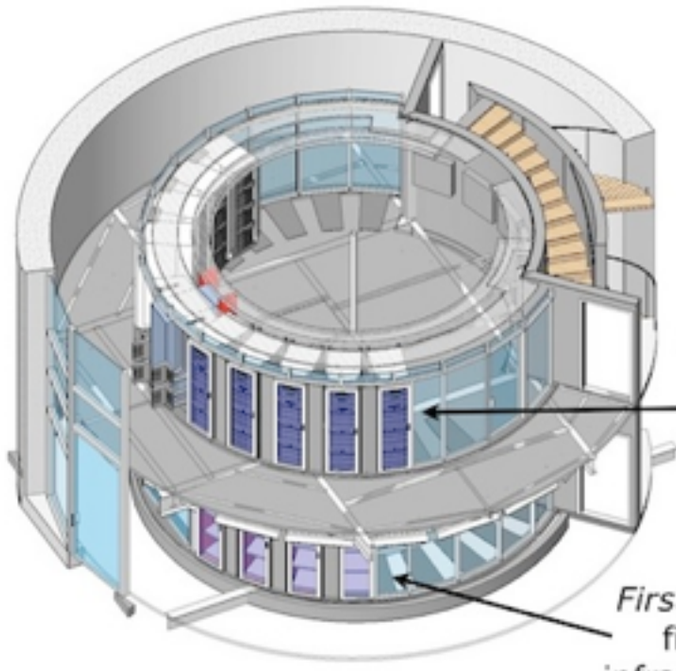


Super-calculateur

- Description:
  - Un ordinateur (super-ordinateur) comprenant des centaines ou des milliers de processeurs ou de microprocesseurs autonomes et indépendants qui calculent en parallèle. Architecture intégrée, pouvant souvent être vue comme un seul OS.
  - MPP: “Massively Parallel Processing”
- Avantages:
  - Permet de mettre en parallèle les ressources de plusieurs ordinateurs souvent identiques afin de réduire le temps nécessaire afin d’effectuer une tâche ou un calcul complexe.
- Limites:
  - Le coût!!!
  - L’espace.
  - Les limites intrinsèques à l’exécution en parallèle d’instructions.
  - Topologie et interconnections des nœuds.
  - Gestion des erreurs.

Super-calculateur

- 960 noeuds de calcul. Chaque noeud comprend:
  - deux processeurs Intel Xeon. Chaque processeur comprend:
    - 4 coeurs
    - 8 Mo cache partagée par les 4 coeurs
- Total:
  - 7680 coeurs
  - 24 To (téra-octet, soit 1,000 Go) mémoire
  - 2 x 500 To (1 Po) de stockage
  - Puissance de calcul: 77 Tflops (opérations à point flottant par seconde)



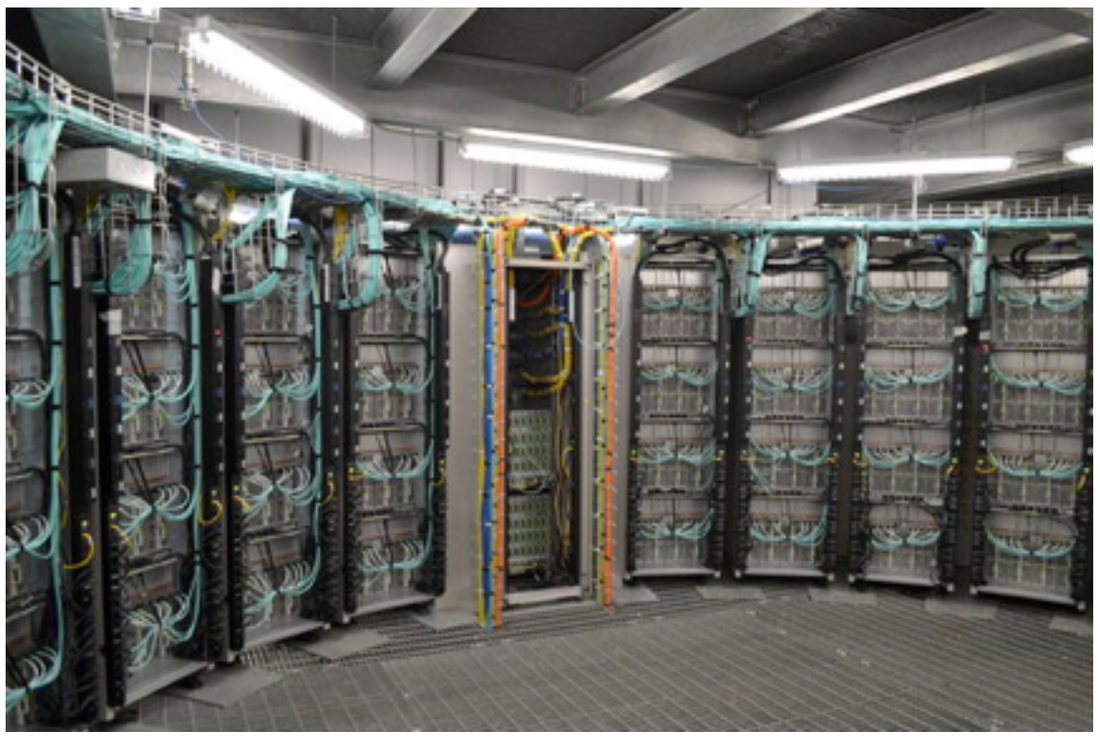
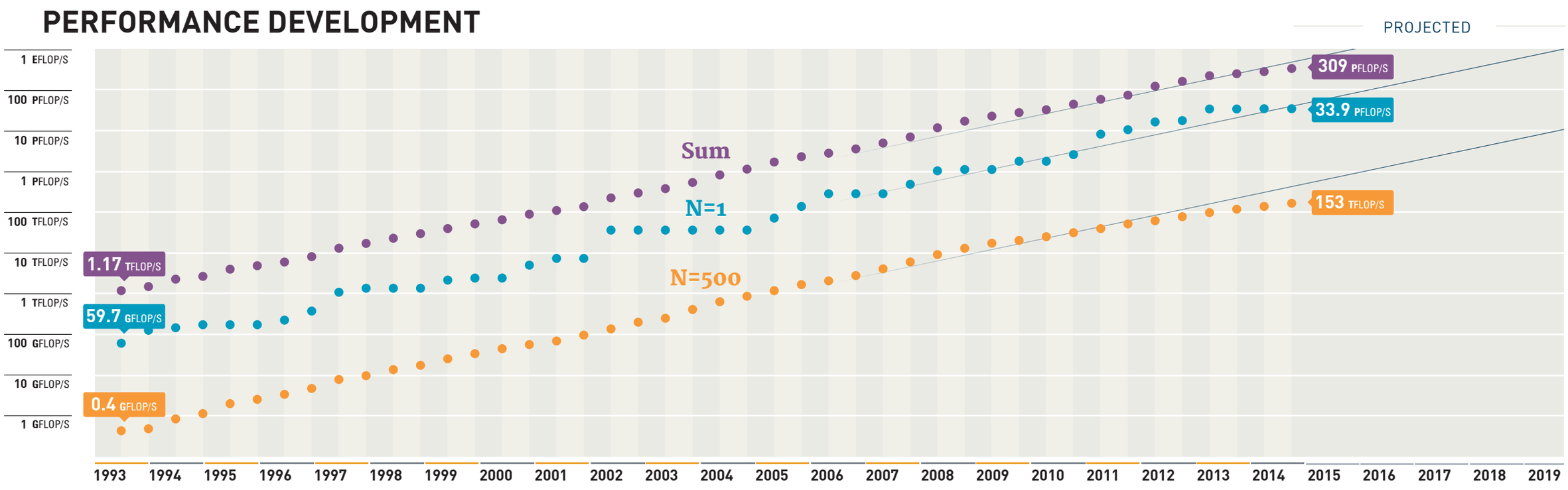


Photo: Hugo Prévost, pieuvre.ca

Source: http://www.calculquebec.ca/fr/access-aux-ressources/serveurs/colosse

Super-calculateurs



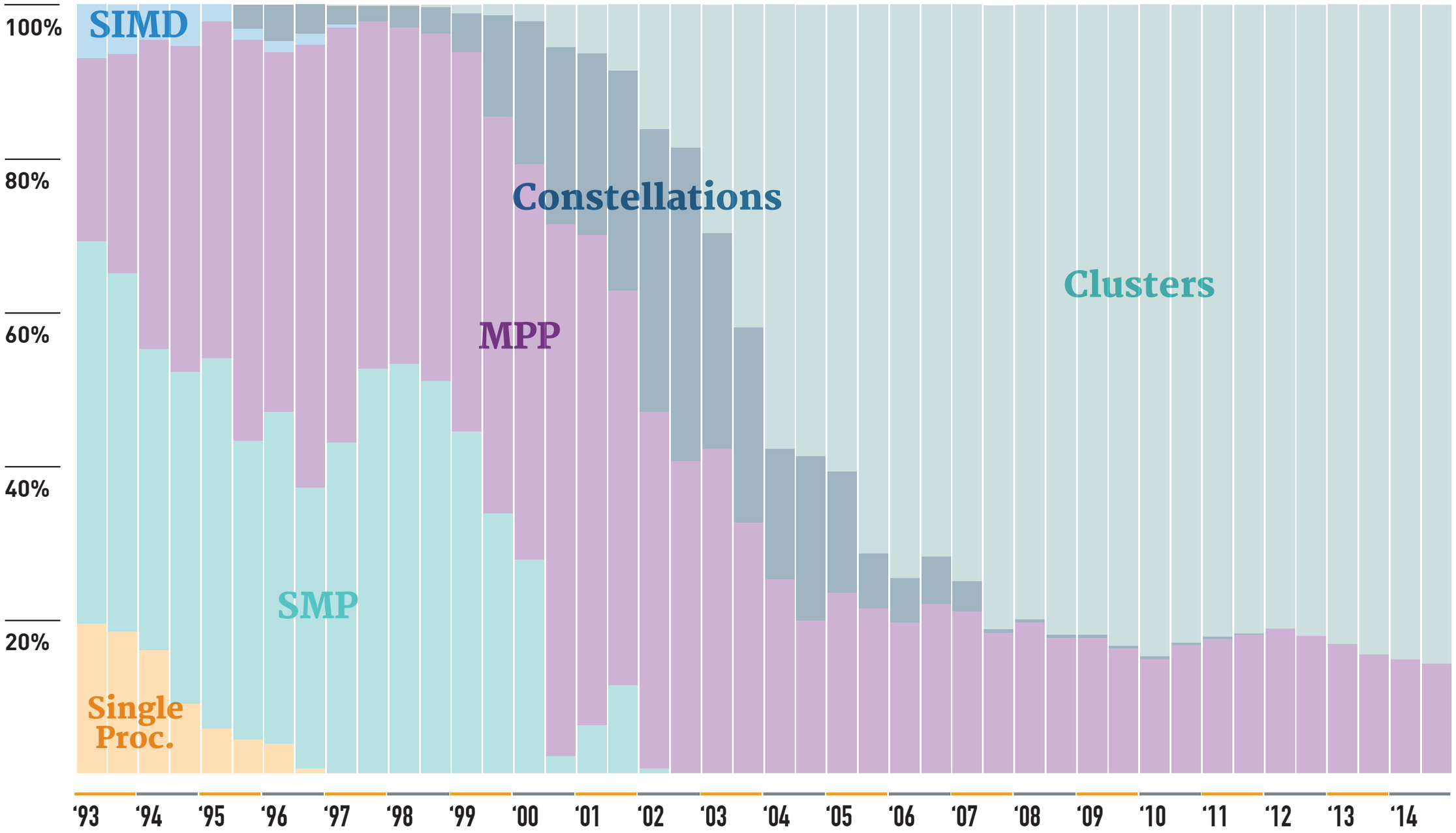
Source: top500.org

Grappe (“cluster”) d’ordinateurs

- Description: Regroupement d’ordinateurs indépendant et hétérogènes partageant une application afin d’accomplir une tâche.
- Chaque processeur possède sa propre mémoire et son propre système d’exploitation. Les processeurs s’échangent de l’information via un protocole de messages
- Avantages:
  - Permet d’obtenir une puissance de calcul très grande à partir d’éléments désuets, disparates et/ou peu dispendieux.
- Limites:
  - Le coût!!!
  - L’espace.
  - Les limites intrinsèques à l’exécution en parallèle d’instructions.
  - Topologie et interconnections des nœuds.
  - Gestion des erreurs.

Super-calculateurs

ARCHITECTURES



Résumé: historique

- Définition: Une architecture parallèle permet l'exécution de deux tâches différentes (ou plus!) simultanément.
- Historique pour ordinateurs de bureau:
  - Les premiers ordinateurs exécutaient les instructions d'un programme unique en séquence. Ces ordinateurs ne faisaient pas de traitement parallèle.
  - Au début des années 1990, l'exécution des instructions d'un programme se fait en parallèle. Les pipelines d'instructions deviennent un élément de base des ordinateurs personnels.
  - Peu après, vers 1993, les microprocesseurs de bureau deviennent superscalaires: des pipelines d'instructions sont exécutées en parallèle.
  - Entre 1993 et 2000, l'exploitation du parallélisme entre les instructions d'un programme est maximisée.
  - Quand la complexité matérielle requise pour exploiter le parallélisme à l'intérieur d'un seul programme explose, les designs de microprocesseurs changent: un microprocesseur exécute des instructions venant de plusieurs processus (2000-2002, multithreading)
  - L'exécution parallèle d'instructions provenant de plusieurs programmes conduit à l'apparition de microprocesseur ayant plusieurs cœurs (2002-aujourd'hui).

Résumé: architectures

- Dans toutes les architectures parallèles, de la mémoire est partagée afin d'échanger des informations entre les différents processeurs.
- Afin d'exécuter des instructions en parallèle et de maximiser l'utilisation des ressources, certaines ressources sont partagées.
- Le niveau de la mémoire partagée et la nature des ressources partagées définissent souvent le nom de l'architecture parallèle:
  - **Pipeline d'instruction**: Les différentes sections de l'unité d'exécution du processeur sont partagées pour exécuter plusieurs instructions en parallèle.
  - **Multithreading**: Les unités d'exécution des pipeline d'instructions sont partagées entre les instructions de différents threads.
  - **Symmetric Multi Processors** (SMP): La mémoire de l'ordinateur est partagée par plusieurs microprocesseurs afin d'exécuter plusieurs threads en parallèle.
  - **Non-Uniform Memory Access** (NUMA): Les entrées/sorties de l'ordinateur, certains bus et, habituellement, les espaces d'adresse, sont partagés afin d'exécuter plusieurs threads en parallèle. En d'autres mots, le système d'exploitation est partagé afin de permettre l'exécution de tâches en parallèle.
  - **Grappe d'ordinateur ou super-ordinateur**: Une application est partagée afin de gérer les threads à exécuter.

Références

- Références
  - Stallings, chapitres:
    - 14.4 (pipelines)
    - 16.1 (architectures superscalaires)
    - 17.2 (SMP)
    - 17.4 (multithreading)
    - 17.6 (NUMA)
  - Englander, chapitres:
    - 8.2 (architectures superscalaires)
    - 11.4 (supercalculateurs)
  - Dongarra et al., “High Performance Computing: Clusters, Constallations, MPPs, and Future Directions”.
    - Accessible via la bibliothèque: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1401802