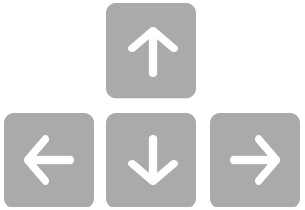
Systèmes Distribués et Répartis

Olivier Lemer

Reveal.js slides

H Hide les notes

Esc Voir toutes les slides

 Naviguer dans les slides

? Liste des raccourcis clavier

Logistique

Horaires
Cours - mardi, 14h45-16h15, en J04.

Labos - mercredi, 13h00-14h30, en B23.

Notation
- 2 tests écrits (67%)
- 4 labos notés (33%)

Matériel du cours
Centralisé sur sdr-classroom.github.io.

Labos
Données et rendus sur GitHub.

Discussions
Feedback, questions, suggestions sur Teams.

Labos

4 semaines par labo, par groupe de deux.

Quiz individuel en fin de chaque labo

Prouvez-nous que vous comprenez votre solution et ses choix.

1. Reliable Broadcast et Mutex — Sans pannes

2. Élection — Avec pannes

3. Synchronisation — Avec pannes

4. Consensus — Avec pannes

Les IAs

Ma vision

Ce qui fera de vous de bon.ne.s ingénieur.e.s

Transformer un besoin en abstractions et leurs interactions
(i.e. définir les boites noires, et comment elles intéragissent)

Réaliser une abstraction
(i.e. comment la boite noire fonctionne pour satisfaire sa spécification)

Réaliser une logique
(i.e. implémenter un algorithme donné)

Utiliser des if et des boucles
(code linéaire simple, niveau introductif)

Utiliser la bonne syntaxe
(dépend du langage, peu de réflexion)

Compétences des LLM

Ça tombe bien

Ce qui vous fera sortir du lot, c'est ce que vous ne pourrez pas déléguer aux IAs.

Les IAs

Votre objectif

Cherchez à sortir du lot.

Construisez-vous une valeur ajoutée aux vibe-coders.

Être capable de dire

- *"Non, cette fonction devrait retourner une promesse, parce que..."*
- *"Non, cette abstraction a trop de responsabilités, il faut..."*
- *"Non, cet import est superflu, utilisons plutôt..."*
- *"Attention, on commence à dépendre de détails d'implémentation..."*
- *"Attention, ce module fait une supposition sur celui-ci qui..."*
- *...*

Programme

1. Introduction - *Définitions, fiabilité, diffusion et pannes*
2. Estampilles - *Horloges logiques, exclusion mutuelle*
3. Jetons - *Exclusion mutuelle*
4. Diffusion - *Élection de leader*
5. Sondes et échos - *Exemples et synchroniseurs*
6. Synchronisation et battements - *Exemples*
7. Consensus (!)

Aujourd'hui

Cours

Définition d'un système distribué et réparti

Classes de fiabilité

Reliable Broadcast (Diffusion Fiable)

Si le temps le permet : Définition des pannes

↓ Système distribué ?

Définition(s)

Système

Réparti

Système s'executant sur

- un ensemble de processus (process, machine, etc),
- sans mémoire partagée,
- vu par l'utilisateur.rice comme une seule entité.

S'emploie plus quand on parle **des taches et leur répartition**.

Distribué

S'emploie plus quand on parle de l'**architecture du système**.

Relativement interchangeables.

Décentralisé

Système dans lequel **il n'existe pas d'autorité centrale** responsable du contrôle du système.

Enjeux

Logiciel

- Problèmes simples deviennent compliqués
- Tous langages ne sont pas adaptés
- Niveau de transparence sur l'aspect réparti

Fiabilité

Résilience à

- Délai du réseau
- Perte de messages
- Crash de machines

Partage de données

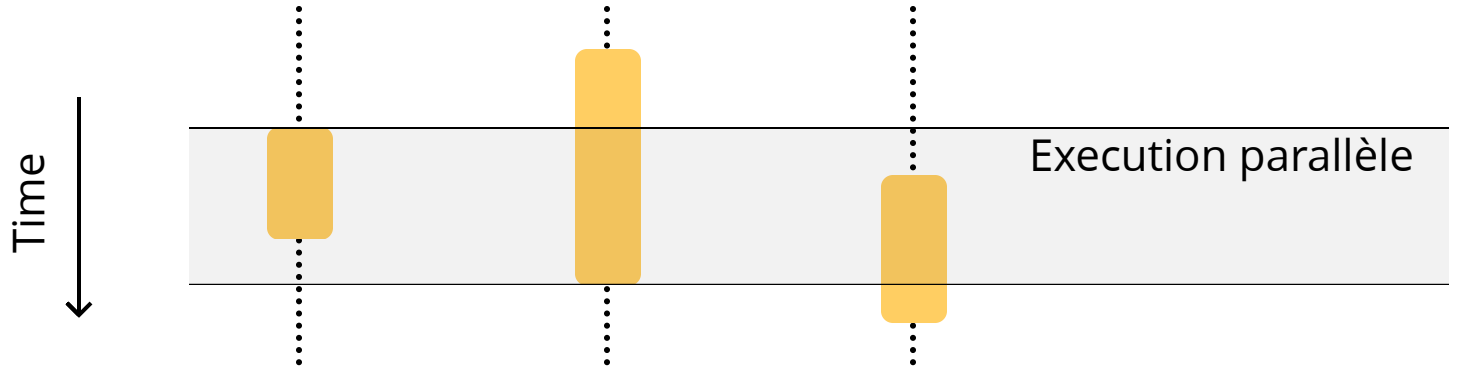
- Synchronisation des données entre machines
- Protection et sécurité des données

↓ Parallèle vs. concurrent

Parallélisme

Définition

Lorsque deux taches sont en cours d'execution *au même instant*.



Question : Quelles unités de traitement executent ces taches ?

Threads
→ Système multi-threaded
(e.g. CPU multi-coeur)

Machines
→ Système distribué
(e.g. Réseau de PC interconnectés)

Difficulté : Coordonner les unités de traitement.

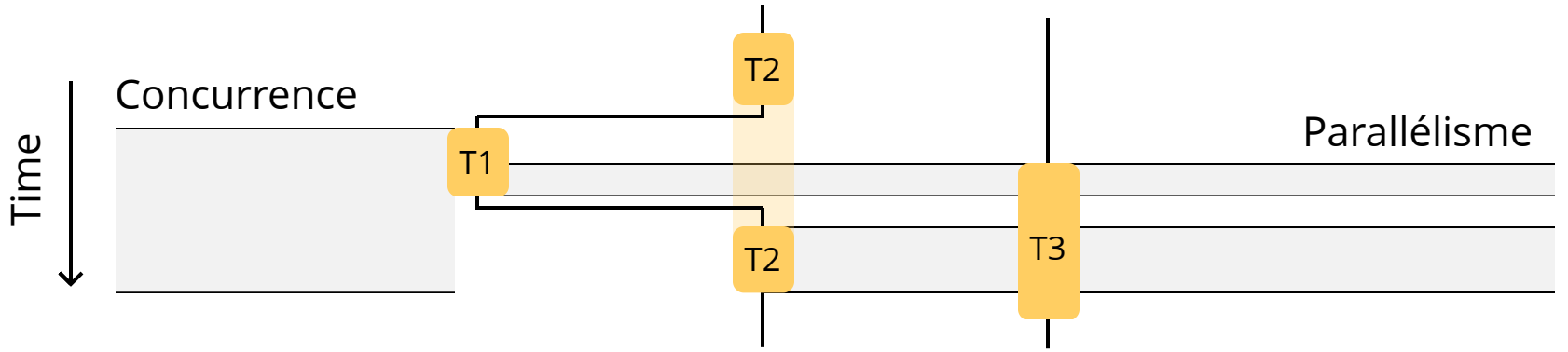
Parallélisme vs. Concurrence

Définition Parallélisme

Lorsque deux taches sont en cours d'execution *au même instant*.

Définition Concurrence

Lorsque deux taches ont progressé dans un interval commun.



→ T1, T2 et T3 s'exécutent de manière concurrente, mais pas toujours parallèle.

↓ Classification de Flynn

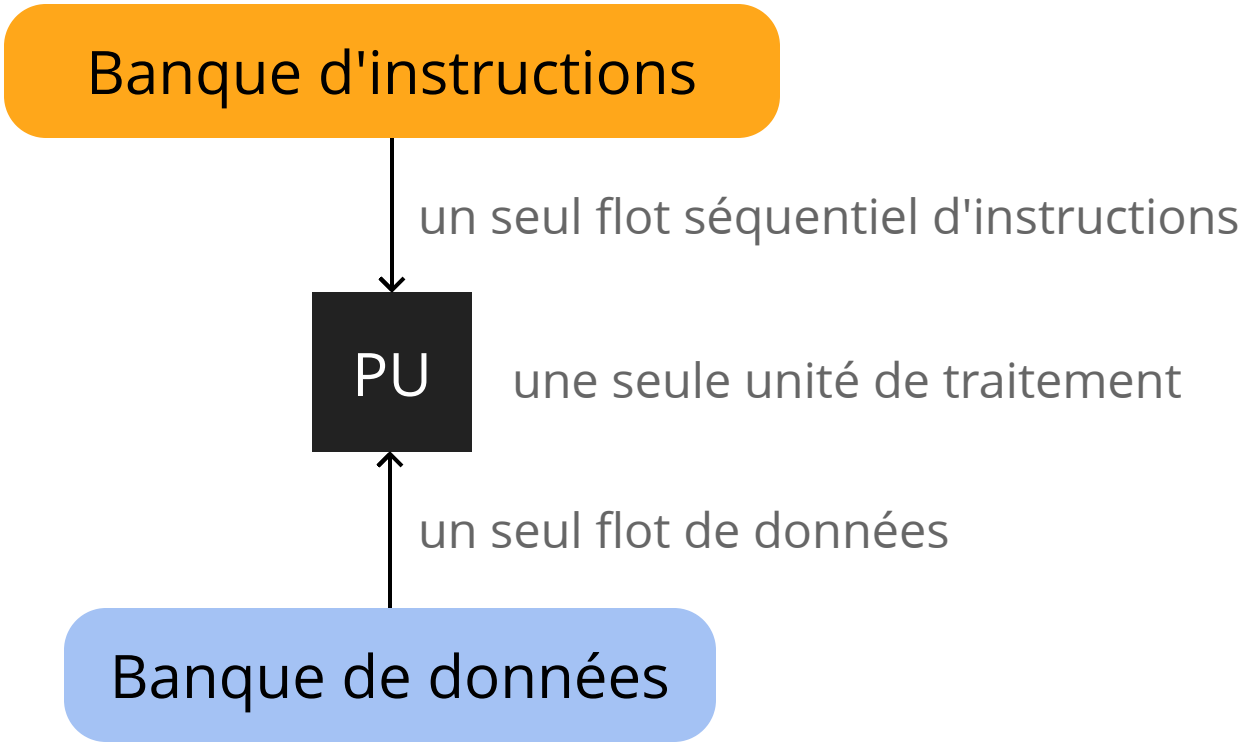
Classification de Flynn

Catégorisation des machines selon 2 axes : flots de **données**, et flots d'**instructions**.
(i.e. contrôle)

Flot de **D**onnées \ Flot d'**I**nstructions *(Contrôle)*	**S**ingle	**M**ultiple
Single	SISD	MISD
Multiple	SIMD	MIMD

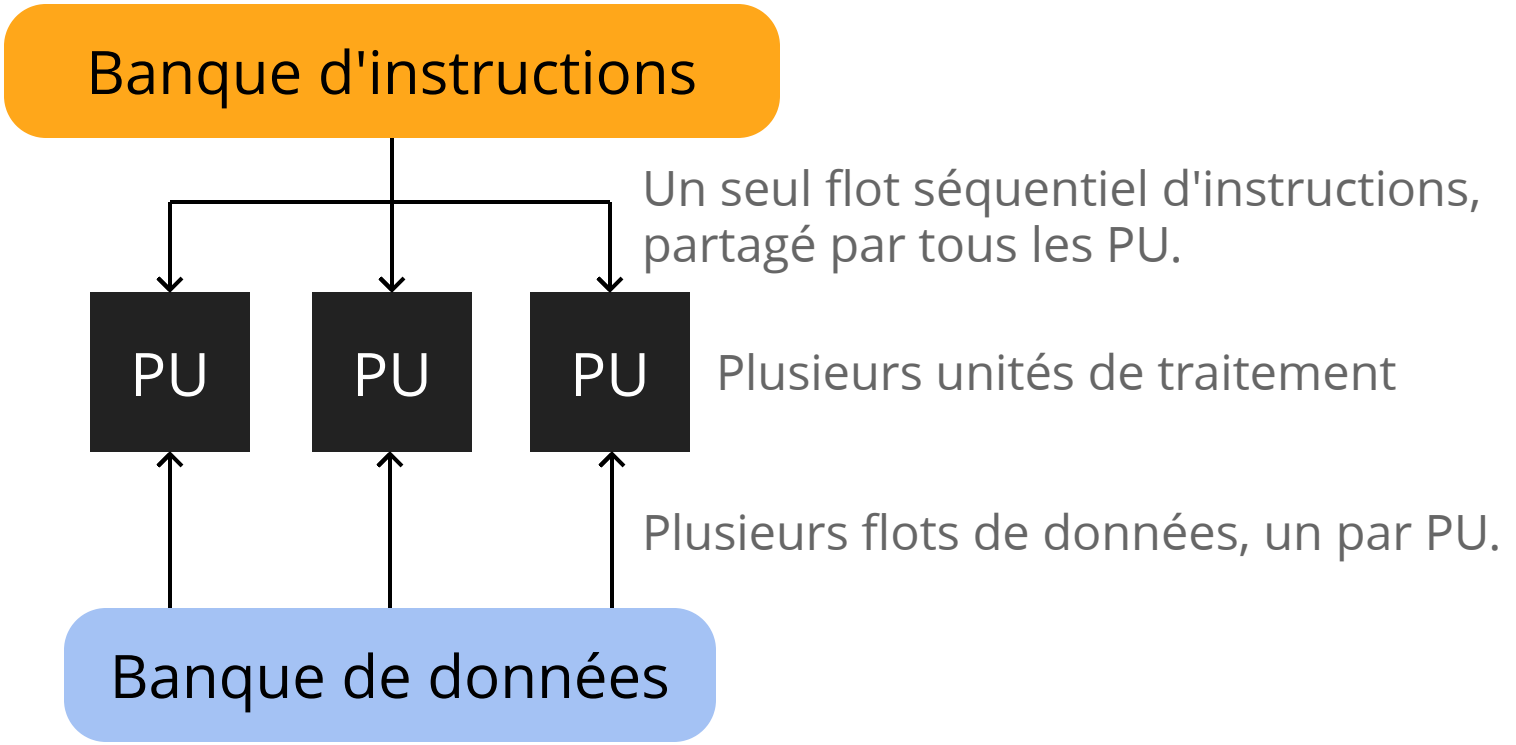
Classification de Flynn

SISD *(a.k.a. Architecture Von Neumann (1945))*



Classification de Flynn

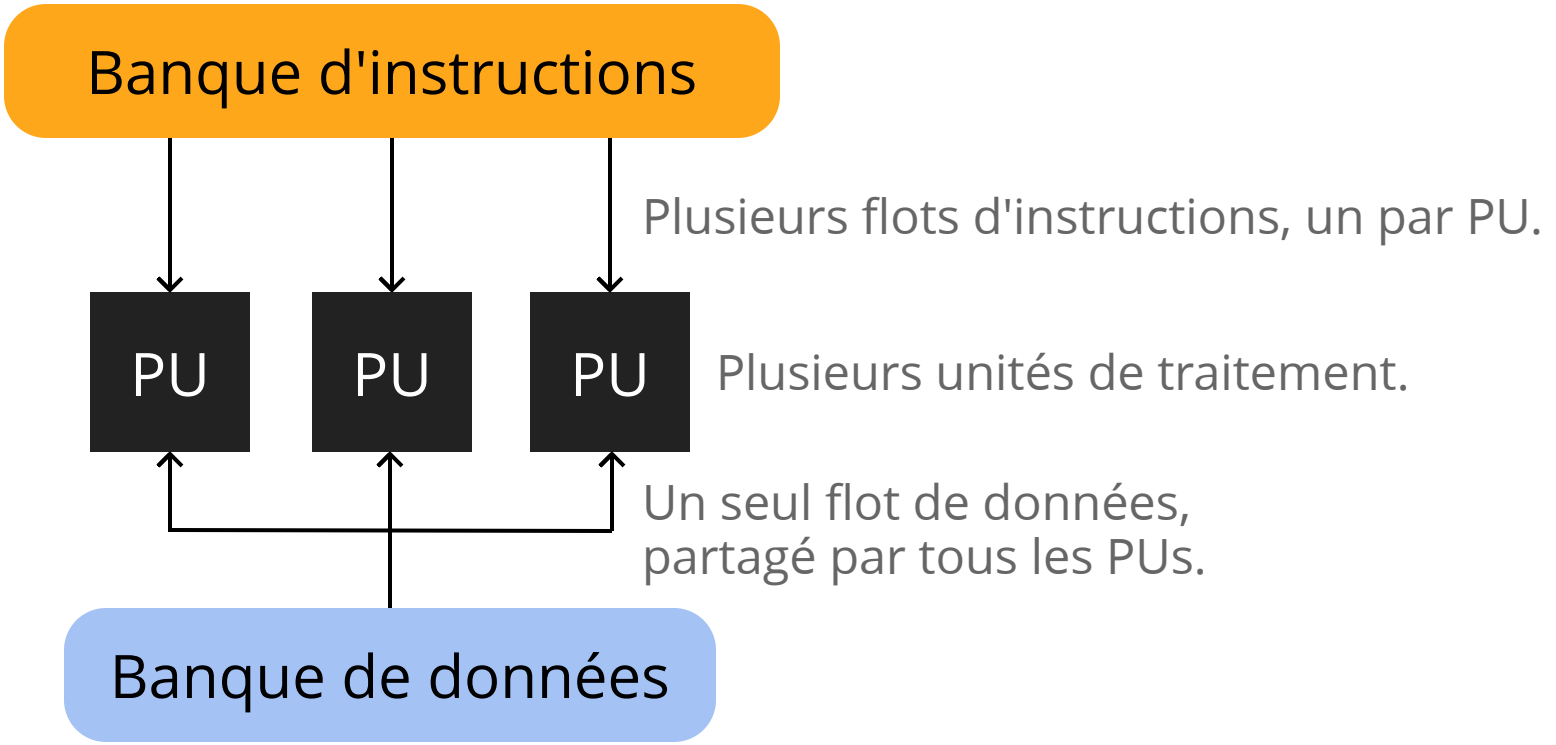
SIMD



Par exemple pour calcul scientifique (vecteurs et matrices)

Classification de Flynn

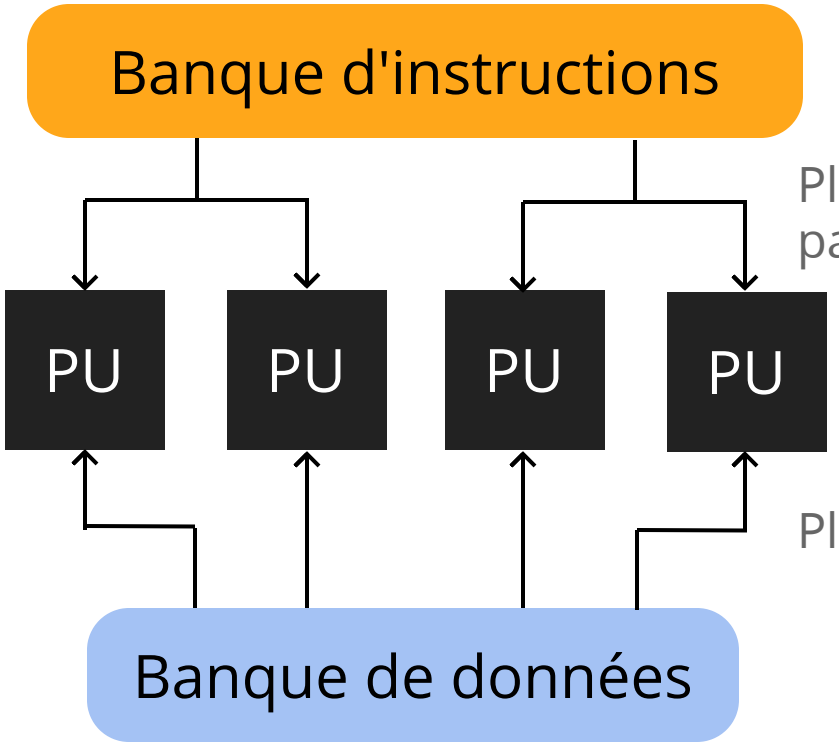
MISD



Architecture théorique...

Classification de Flynn

MIMD



Plusieurs flots d'instructions,
parfois partagés par plusieurs PUs.

Plusieurs unités de traitement

Plusieurs flots de données, un par PU.

L'exécution peut ici être asynchrone. L'enjeu est la synchronisation des PUs.

Classification de Flynn

Catégorisation des machines selon 2 axes : flots de **données**, et flots d'**instruction**s.

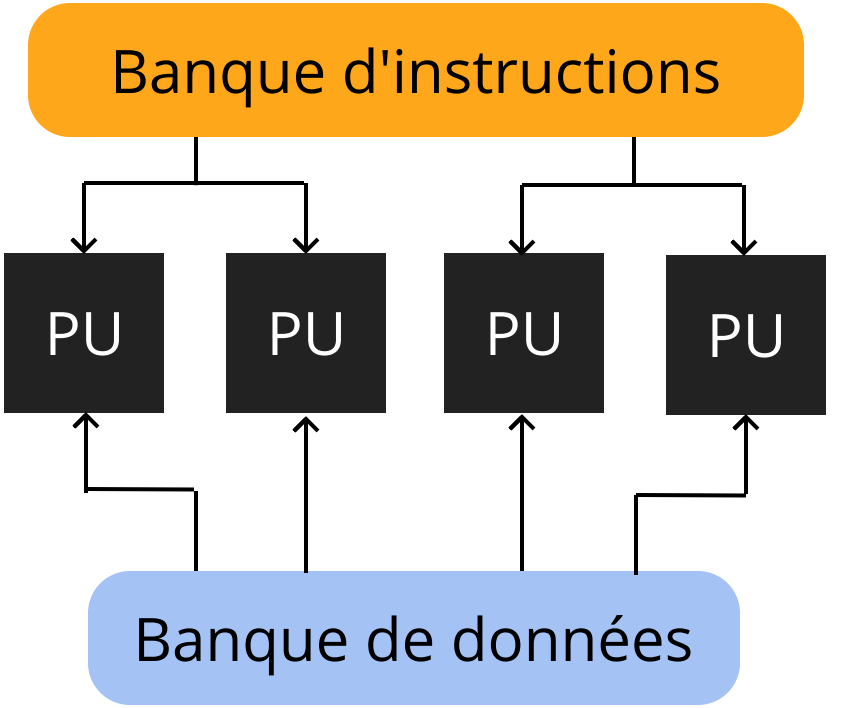
(i.e. contrôle)

Flot de **D**onnées \ Flot d'**I**nstructions *(Contrôle)*	**S**ingle	**M**ultiple
Single	SISD	MISD
Multiple	SIMD	Shared memory MIMD Distributed memory

Classification de Flynn

MIMD (Shared Memory)

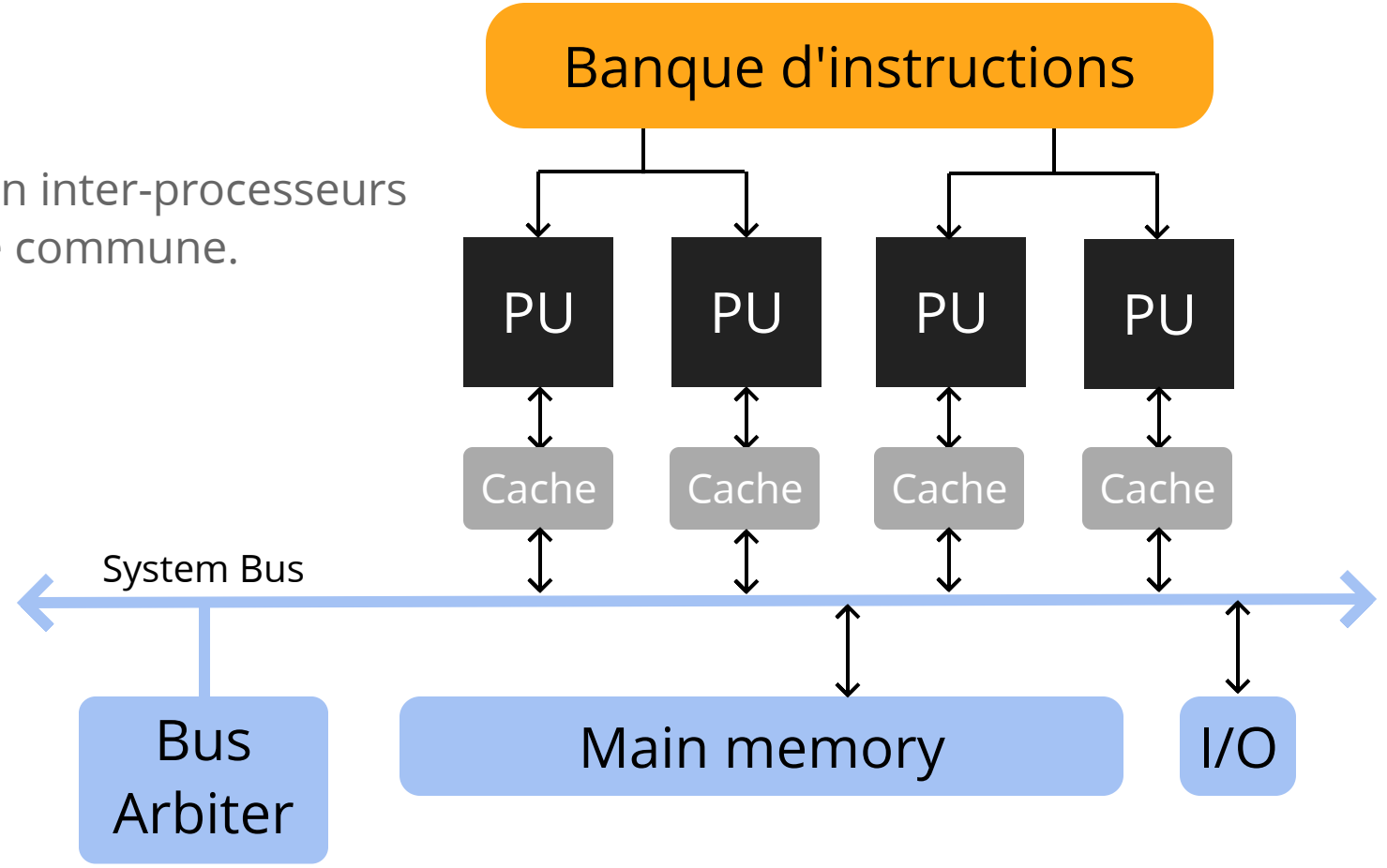
Communication inter-processeurs via la mémoire commune.



Classification de Flynn

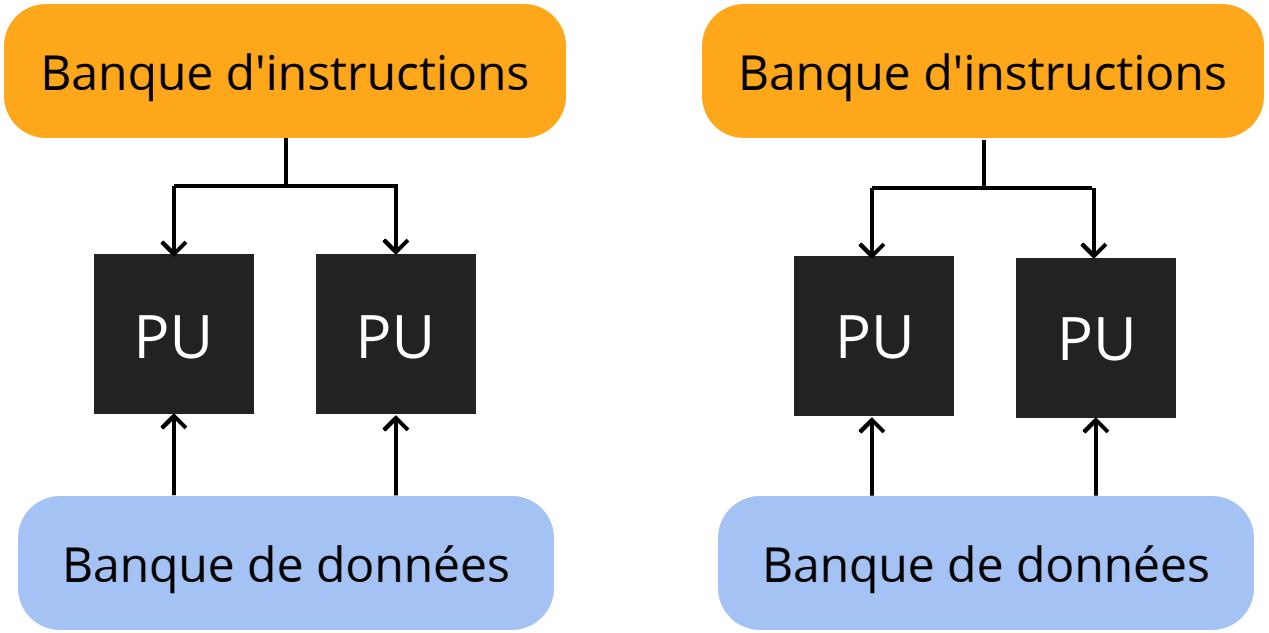
MIMD (Shared Memory)

Communication inter-processeurs via la mémoire commune.



Classification de Flynn

MIMD (Distributed Memory)

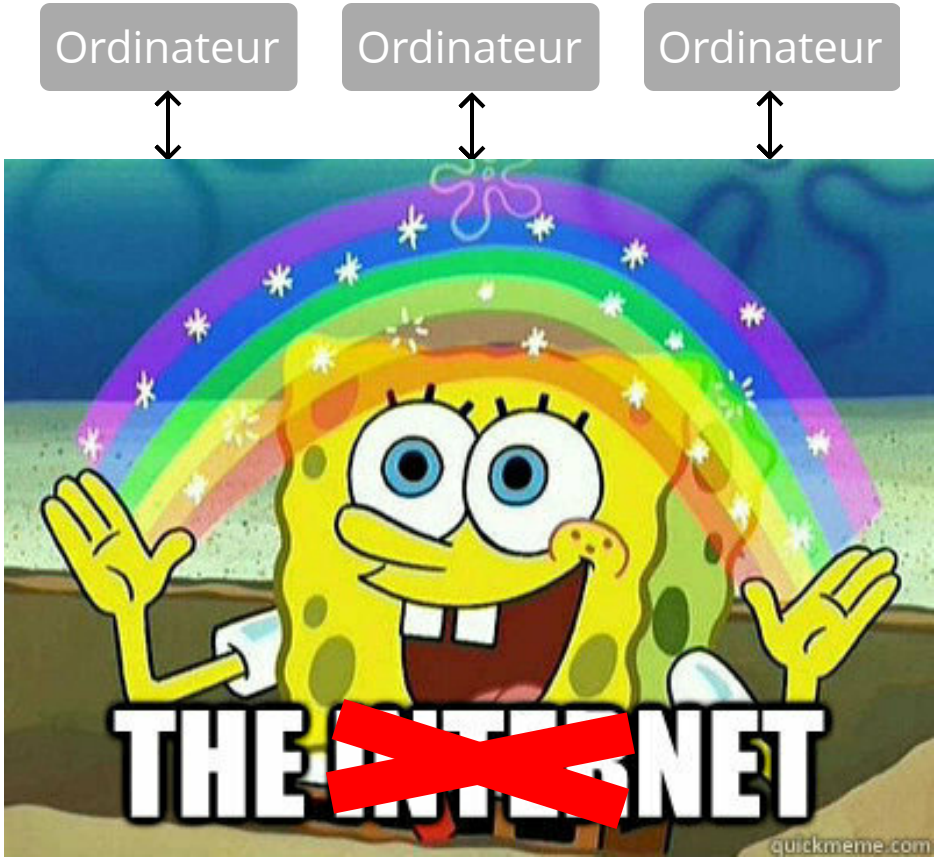


Communication inter-processeurs via le réseau.

Classification de Flynn

MIMD (Distributed Memory)

(e.g. Massively Parallel Processing (MPP))



Plusieurs ***ordinateurs*** distincts.

Interconnection par réseau.

Par exemple

- Computer clusters
- Grid Computing
- BOINC



Couplage

Matériel vs. logiciel

↓ Couplage matériel

Le couplage matériel

"Quantité et qualité des liens entre éléments."

Couplage fort

Beaucoup de liens, rapides.

Shared Memory MIMD

- *Débit élevé*
- *Délai faible*

Peut partager beaucoup, rapidement.

Couplage faible

Peu de liens, lents.

Distributed Memory MIMD

- *Débit faible*
- *Délai élevé*

Peut partager peu, avec delai.

Ôyez, concepteur.rices !

En fonction du couplage de l'architecture matérielle ciblée, une même application devra être conçue très différemment.

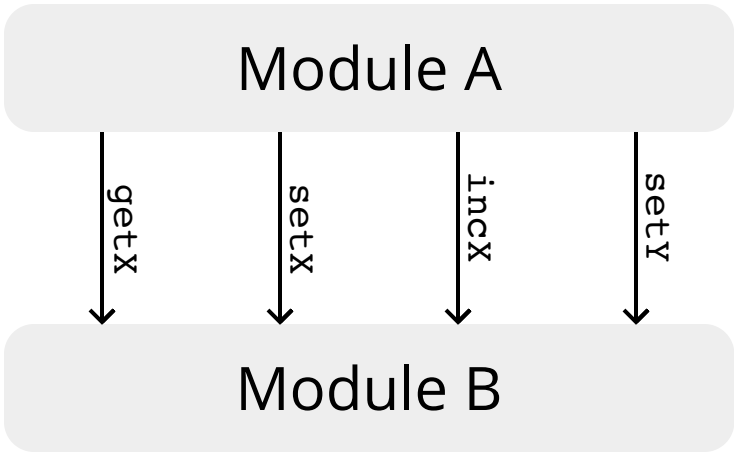
↓ Couplage logiciel

Le couplage logiciel

"Quantité et qualité des liens entre éléments."

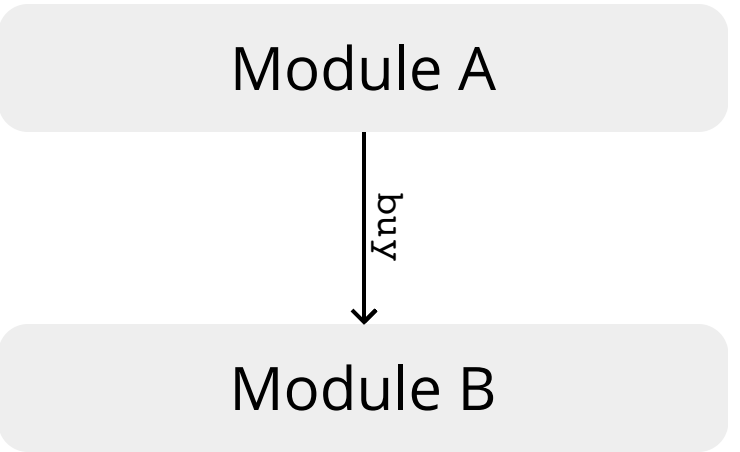
Couplage fort

Beaucoup de liens, rapides.



Couplage faible

Peu de liens, lents.

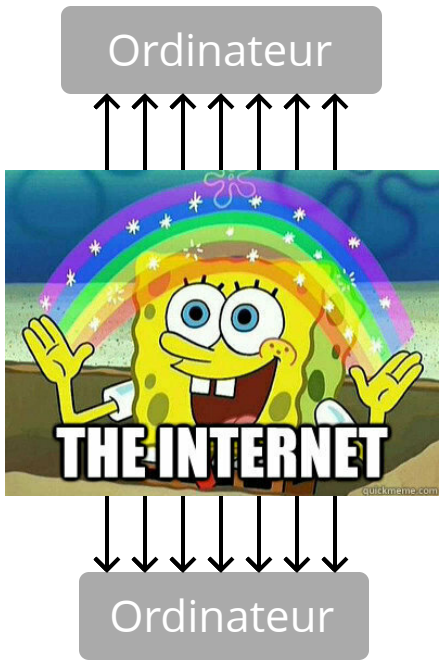


Généralement, on vise un couplage logiciel faible :

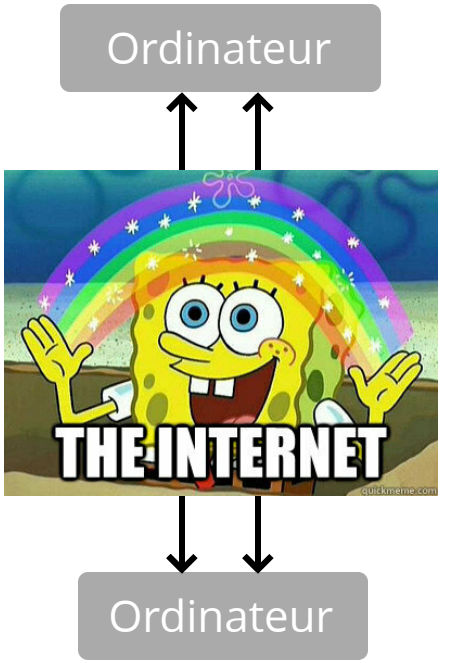
- interfaces plus claires
- risque de bugs moindre

Couplage logiciel et Execution réseau

Execution réseau : couplage matériel faible,
donc coût de communication élevé,
donc couplage logiciel fort "couteux".



vs



Execution réseau vs Programme réparti

Logiciel réseau simple

(telnet, wget, ssh)

- Serveur simple

Logiciel réseau faiblement couplé

(NFS, iCloud Drive)

- Serveur implicitement distribué
- Couplage logiciel naturellement faible

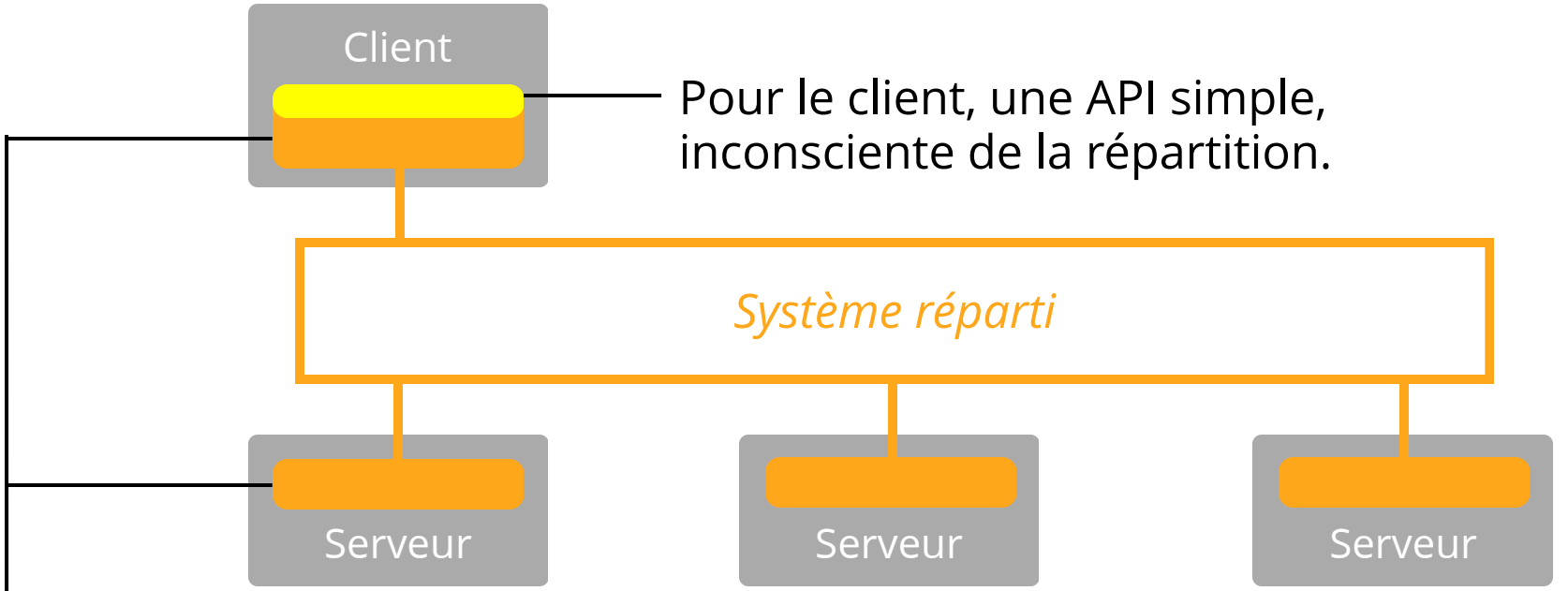
Logiciel réseau réparti

(calcul distribué)

- Serveur implicitement distribué
- Couplage logiciel à tendance forte

  Conception logicielle combat ce couplage

Couche logicielle de répartition



Système réparti

On pourrait donc dire qu'un système réparti est

> l'execution d'une logique nécessitant un **couplage logiciel fort**, sur du matériel limité à un **couplage matériel faible**.

Le challenge est d'optimiser le couplage logiciel effectif pour assurer une performance élevée.



Propriétés

Un *bon* système réparti

Système réparti

Qu'est-ce qu'un bon système réparti ?

1. **Abstraction**

2. **Fiabilité**

3. **Performance**

4. **Dimensionnement**

Système réparti

Qu'est-ce qu'un bon système réparti ?

1. **Abstraction**

    Emplacement des processus et données
    Pas d'adresses physiques des machines ou des données.

    Migration des processus et données
    Déplacement de ressource (processus, données) invisible.

    Duplication des données
    Gestion implicite des copies éventuelles.

    Cohérence des données
    Gestion implicite de la concurrence.

Système réparti

Qu'est-ce qu'un bon système réparti ?

1. **Abstraction**
   (Emplacement, Migration, Duplication, Cohérence)

2. **Fiabilité**

3. **Performance**

4. **Dimensionnement**

Système réparti

Qu'est-ce qu'un bon système réparti ?

2. **Fiabilité**

   Disponibilité
   Résilience aux pannes de machines et de réseau

   Cohérence
   État toujours correct (récupération après panne, résistance aux attaques)

Système réparti

Qu'est-ce qu'un bon système réparti ?

1. **Abstraction**
   (Emplacement, Migration, Duplication, Cohérence)

2. **Fiabilité**
   (Disponibilité, Cohérence)

3. **Performance**

4. **Dimensionnement**

Système réparti

Qu'est-ce qu'un bon système réparti ?

3. **Performance**

Parallélisme maximal
Tirer profit du parallélisme, éviter qu'une machine soit en attente de travail.

Communication minimale
Diminuer le nombre d'échange de messages.

Tradeoff Performance-Fiabilité **:**
Garantir la fiabilité nécessite des protocoles limitant les performances.

Système réparti

Qu'est-ce qu'un bon système réparti ?

1. **Abstraction**
   (Emplacement, Migration, Duplication, Cohérence)
2. **Fiabilité**
   (Disponibilité, Cohérence)
3. **Performance**
   (Parallélisme, Communication)
4. **Dimensionnement**

Système réparti

Qu'est-ce qu'un bon système réparti ?

4. **Dimensionnement** *(scalability)*

   Extensibilité
   Ajouter une machine doit être possible et peu couteux.

   Complexité algorithmique faible
   Avoir plus de machines ne doit pas rendre le service notablement plus lent.

Ceci implique d'éviter les goulots d'étranglement, par exemple

- Élément centralisé
- Nécessité de connaissance d'un état global
  Les algorithmes n'ont donc accès qu'à des informations partielles du système

Système réparti

Qu'est-ce qu'un bon système réparti ?

1. **Abstraction**
   (Emplacement, Migration, Duplication, Cohérence)

2. **Fiabilité**
   (Disponibilité, Cohérence)

3. **Performance**
   (Parallélisme, Communication)

4. **Dimensionnement**
   (Extensibilité, Complexité)