Les Systèmes d'Exploitation

Les mémoires virtuelles

Gérard Padiou

Département Informatique et Mathématiques appliquées
ENSEEIHT

Octobre 2010

plan

La notion de mémoire virtuelle

Origine

Comment confiner les processus à leur espace programme ?

Comment optimiser l'utilisation de la mémoire centrale ?

Propriétés à garantir

- Confiner un processus :
  - Détecter et empêcher tout accès mémoire hors de l'image binaire du programme ;
  - Eviter la modification de code,
  - Eviter la « modification des constantes ».

  ☞contrôle des références mémoires ;
- Partage équitable et optimal de la mémoire centrale ;

Les difficultés originelles de placement en mémoire

Implantation d'une image en mémoire

- $\Rightarrow$ trouver une zone libre suffisamment grande :
  ☞ gestion de zones de taille variable ;
- Chargement à une adresse de début choisie par l'allocateur :
  ☞ translations des adresses absolues dans l'image ;
- Chargement de zones qui ne seront peut-être jamais référencées ;
- Allocation durant toute la durée de vie du processus ;
- Problème d'émiettement des zones de mémoire.

La notion de mémoire virtuelle

Les optimisations réalisées

- Charger un programme toujours au « même endroit » ;
- Eclater l'implantation en mémoire centrale d'un programme ;
- Protéger l'espace d'exécution d'un processus des autres processus (et de lui-même) ;
- Ne charger en mémoire centrale ce qui est(sera) utile ;
- Exécuter des programmes de taille plus grande que l'espace mémoire réel disponible.

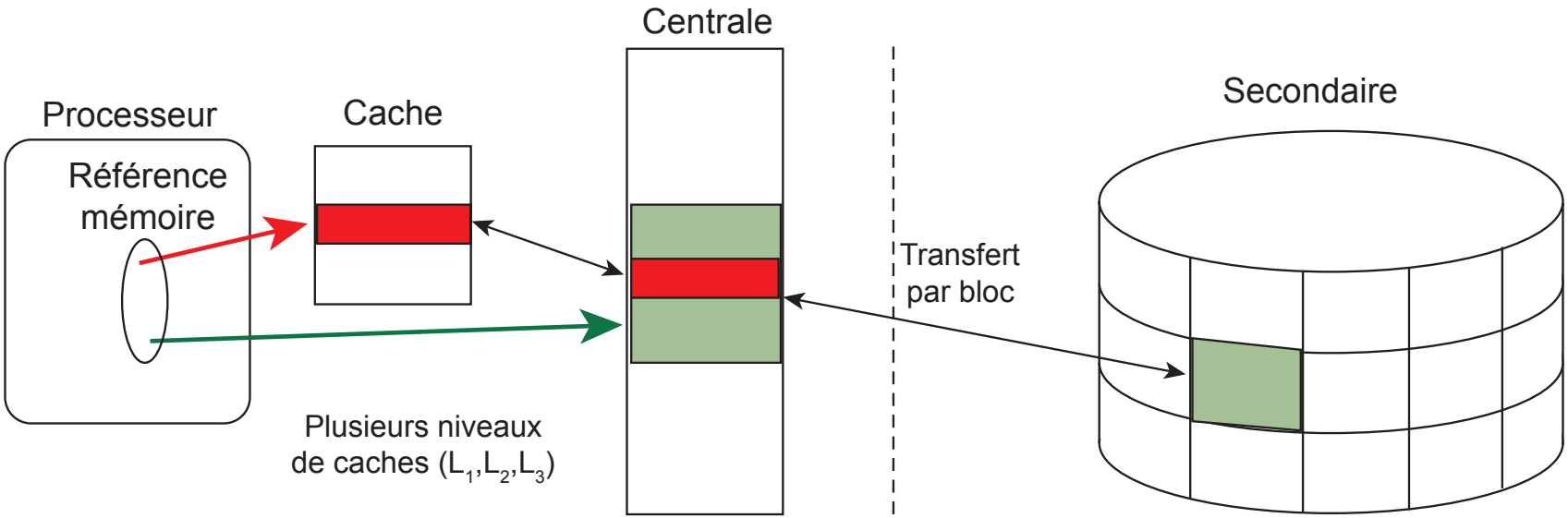
Idée de hiérarchie de mémoires

Les mémoires

- Espace mémoire composé de plusieurs types.
- Propriétés différentes en capacité, temps d'accès, prix.
- Réplication partielle d'information entre types de mémoire.

Type de mémoire	cache	centrale	secondaire
Capacité	Ko	Mo	Go ou To
Rapidité d'accès	1 à 5 ns	50 à 100 ns	accès par bloc
Prix	élevé	moyen	faible
Energie	élevée	moyenne	faible

Idée de hiérarchie de mémoires

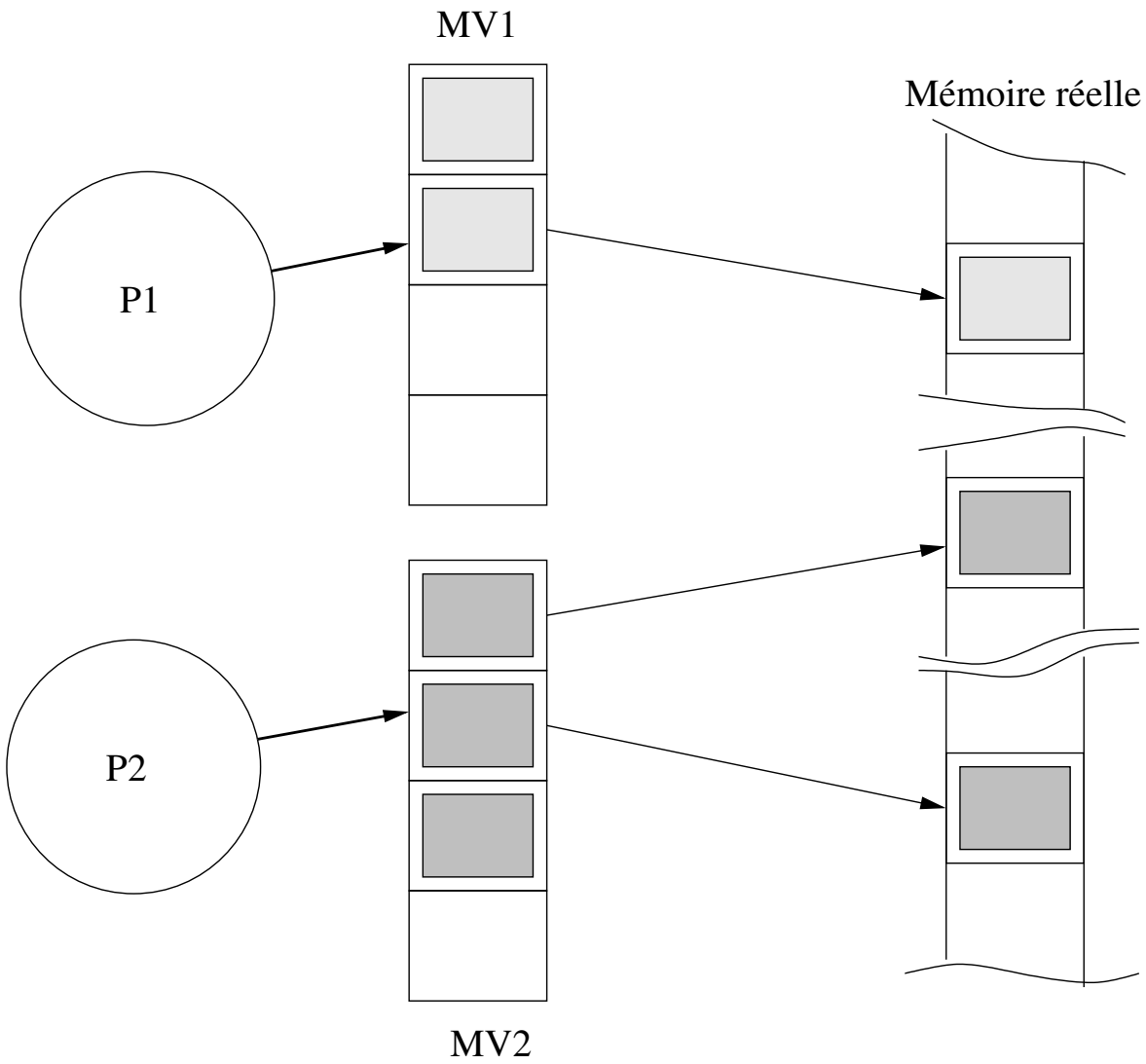


Le contenu de la mémoire

Les différents types d'information

- Code
  - commun (le noyau par exemple)
    - réentrant (exécutable en parallèle par plusieurs processus)
    - critique (exécutable par un seul processus au plus)
  - privé
- Données
  - communes
    - partageables (constantes, accès en lecture)
    - critiques (nécessité de synchronisation)
  - privées

Le principe de base : passage du virtuel au réel



Les unités de gestion mémoire

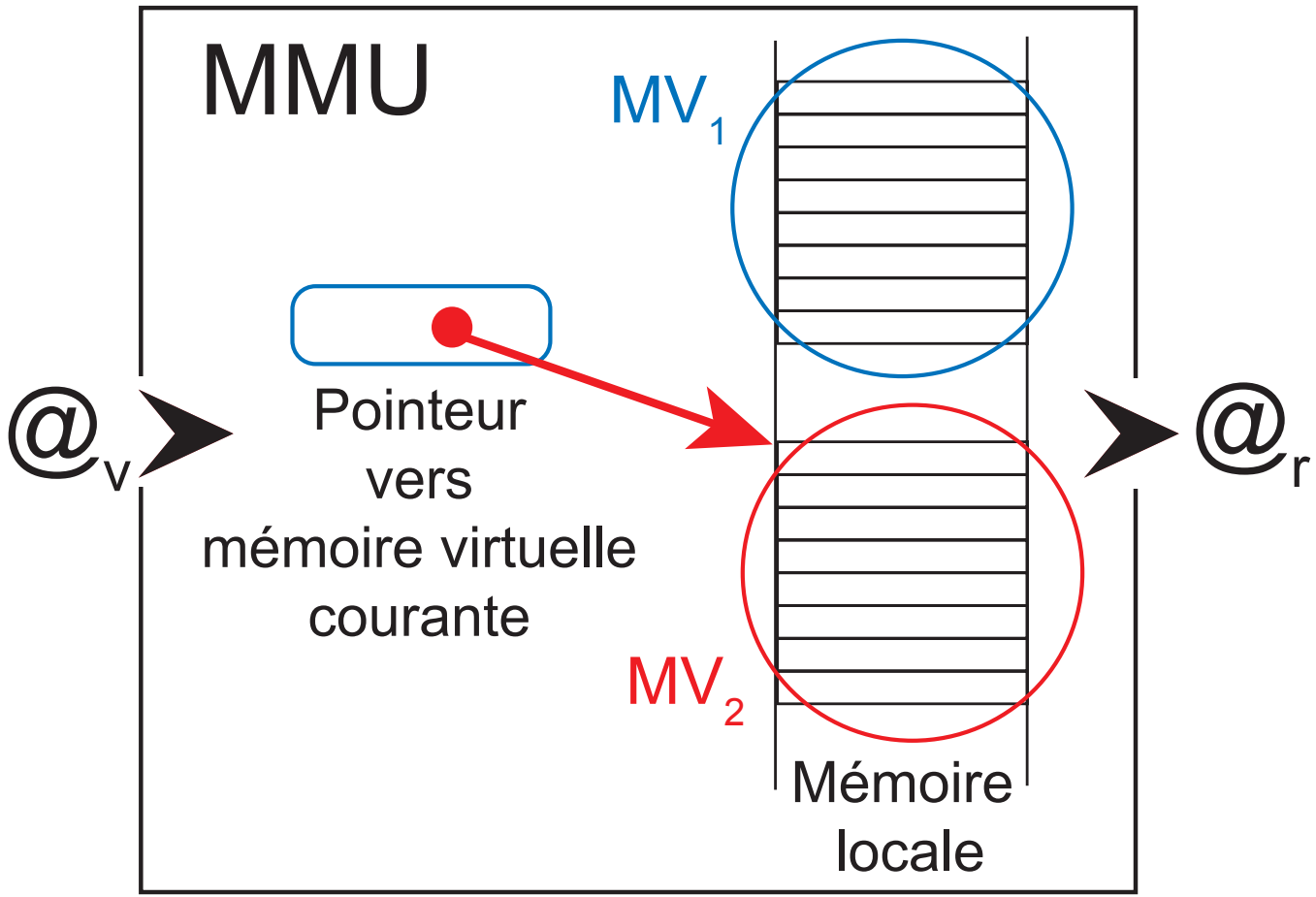
Memory Management Unit (MMU)

Fonctionnalités d'une MMU

- Permettre une conversion d'adresse virtuelle à réelle rapide :
- Détecter les adresses ou accès erronés :
  ☞ Exception «violation de protection mémoire ».
- Permettre d'associer un espace d'adressage privé à chaque image programme ≡ à chaque processus.
- Une MMU possède sa propre mémoire locale pour contenir les descripteurs de mémoire virtuelle.

Les unités de gestion mémoire

Principes de base



Les unités de gestion mémoire

Une entrée de table de pages

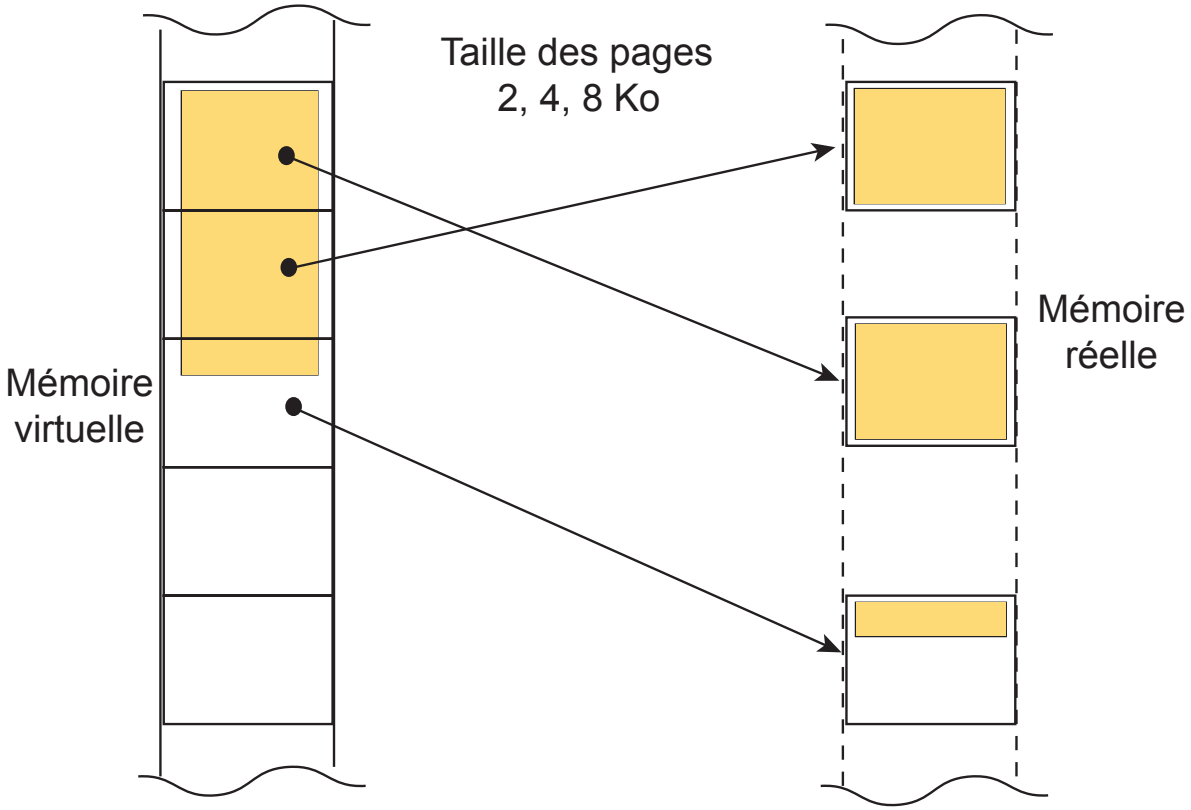
Contenu

- Indicateurs :
  - de protection : r (lecture), w(écriture), x(branchement) ;
  - de présence : P ;
  - pour les systèmes à pagination à la demande :
    - de référence : U (Used Bit) ;
    - d'écriture : D (Dirty Bit).
- Page réelle associée (si présente).

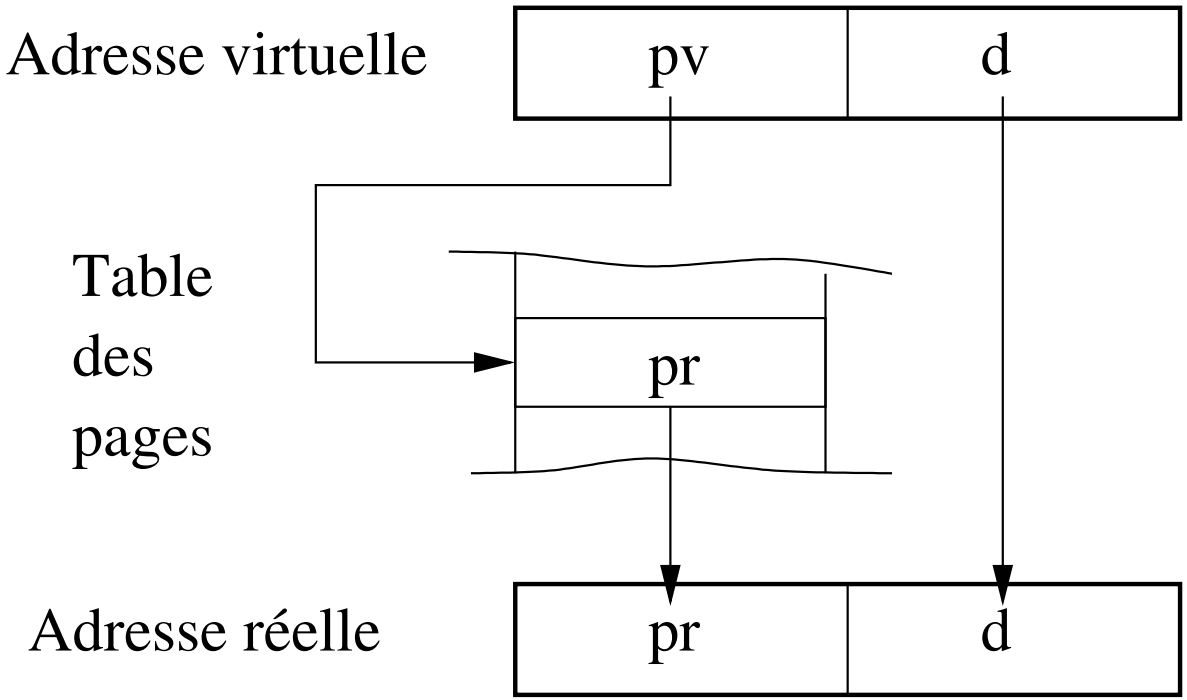
Projection de l'espace virtuel sur l'espace réel

La pagination

- Le type page : typedef $Page = Byte[2^p]$ avec p=10 à 12 ;
- Structure logique mémoire virtuelle : typedef $Mv = Page[2^n]$;
- Structure logique mémoire réelle : typedef $Mr = Page[2^m]$;



L'adressage paginé



La pagination

Idée de base

Même structure logique des deux espaces.

Avantages

- Conversion d'adresse simple ;
- Allocation simple des pages réelles.
- Va-et-vient simple.

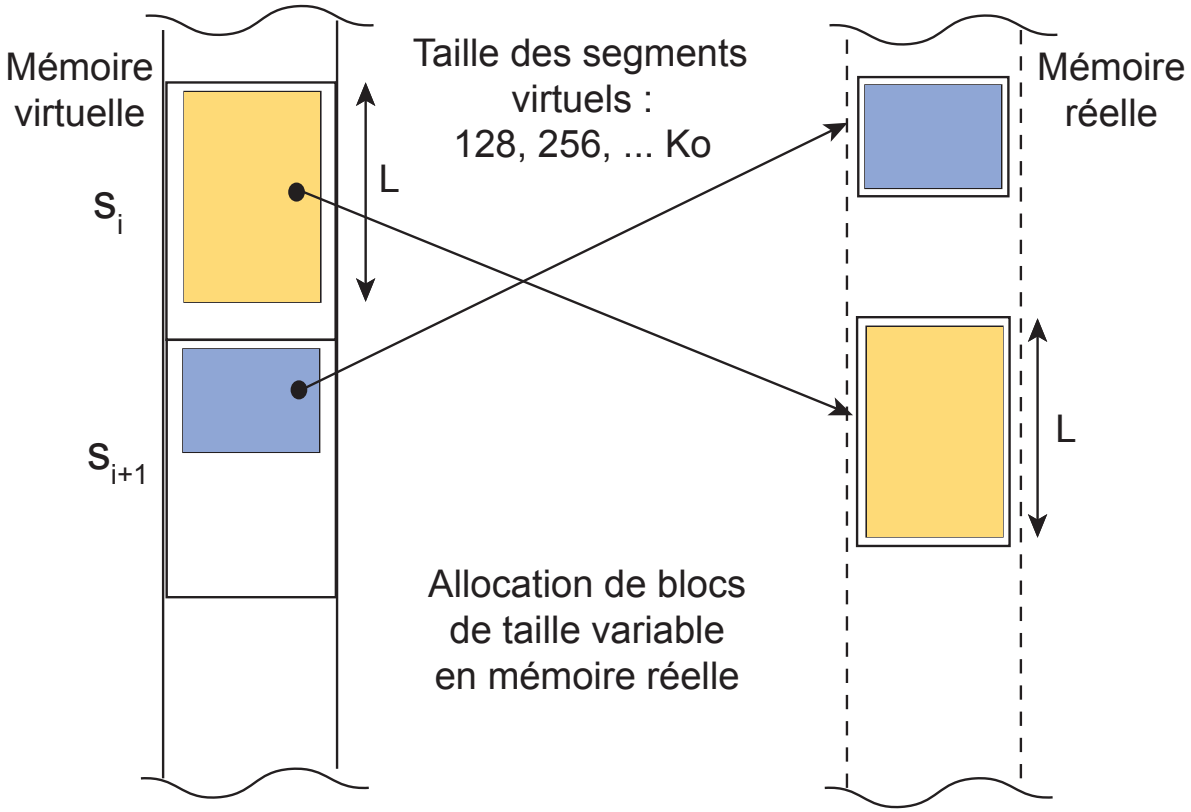
Inconvénients

- Découpage arbitraire en pages des segments de code ou données ;
- Perte d'espace dans les pages de fin de segment.

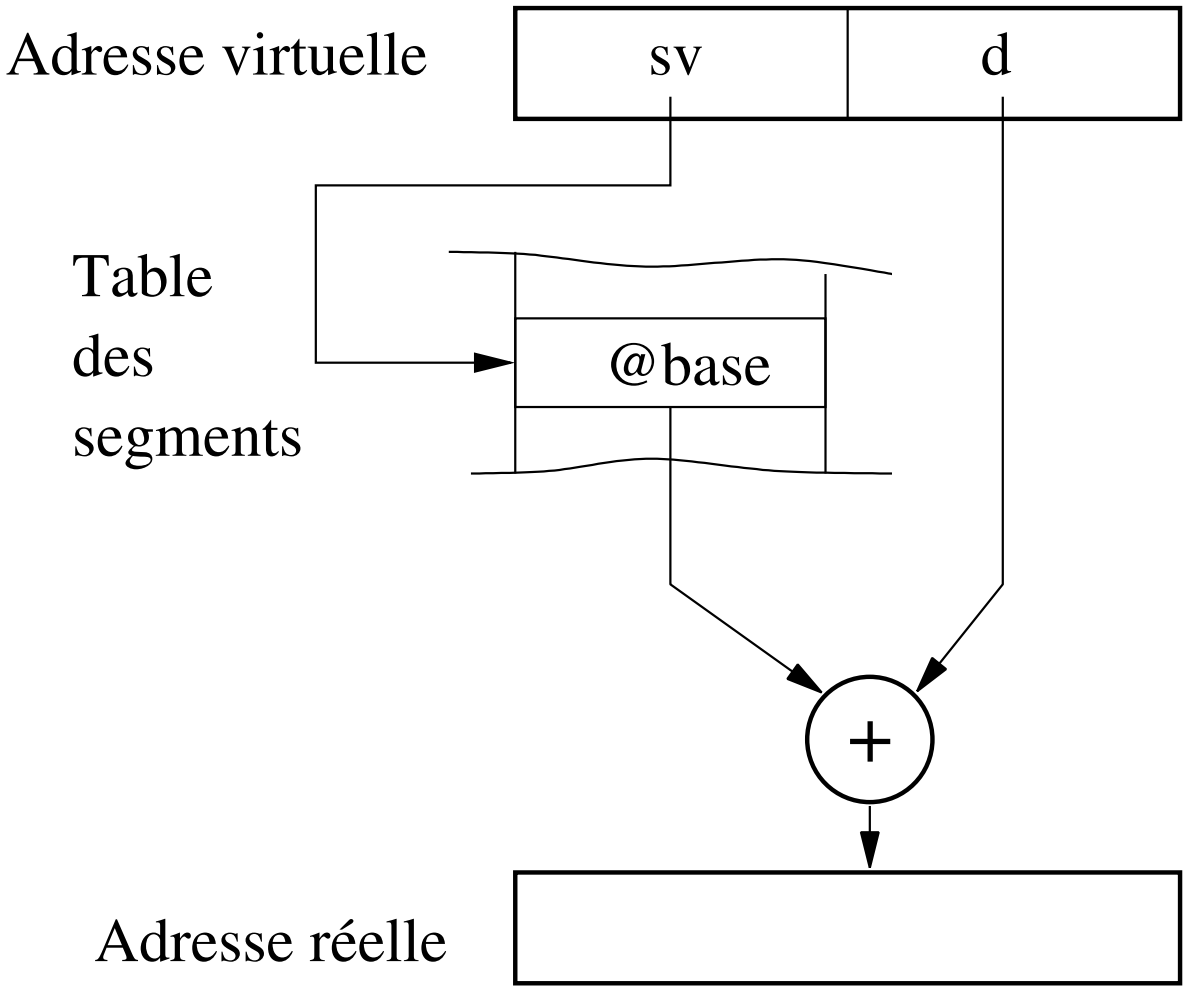
Projection de l'espace virtuel sur l'espace réel

La segmentation

- Le type page : typedef $Segment = Byte[2^p]$ avec $p = 16$ à 18 ;
- Structure logique mém. virtuelle : typedef $Mv = Segment[2^n]$;
- Structure logique mémoire réelle : typedef $Mr = Byte[2^m]$;



L'adressage segmenté



La segmentation

Idée de base

Association d'une section d'image binaire à un segment.

Avantages

- Adaptation à la structure logique du programme ;
- Allocation stricte de l'espace nécessaire.

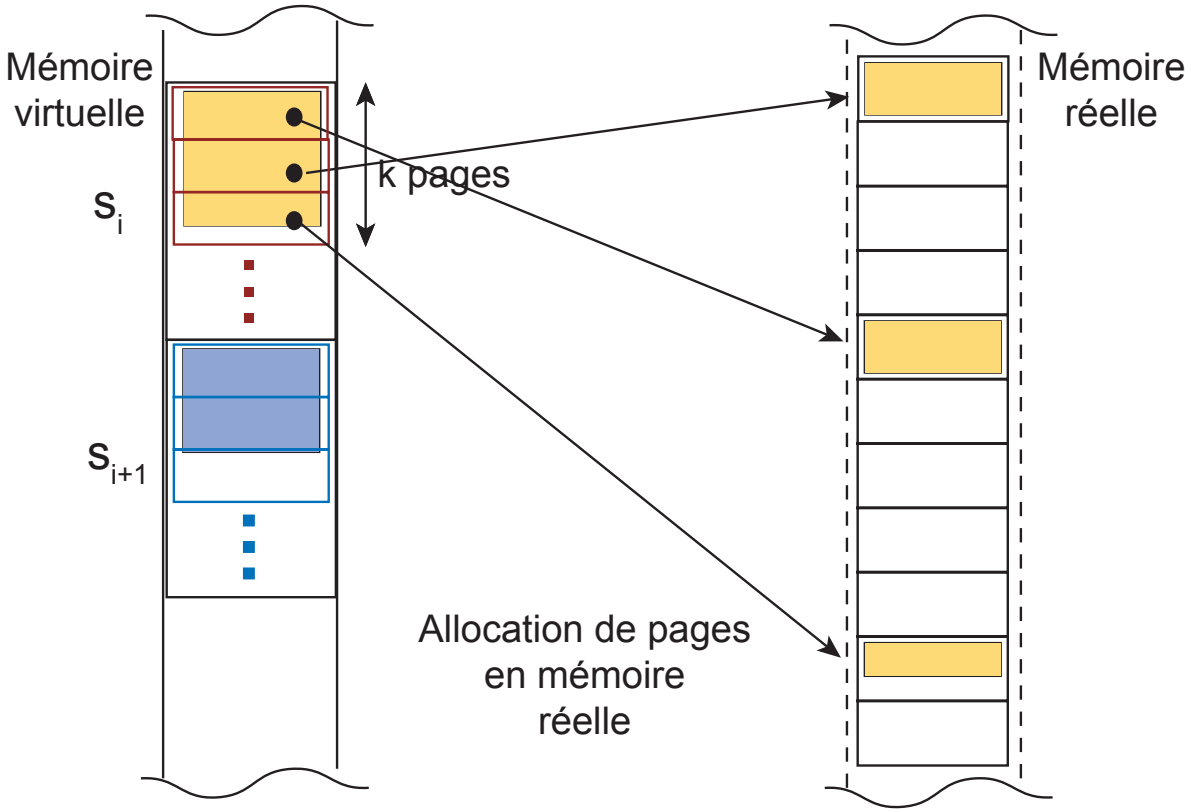
Inconvénients

- Conversion plus complexe ;
- Taille variable des segments ;
- Allocation plus compliquée de zones de mémoire centrale.

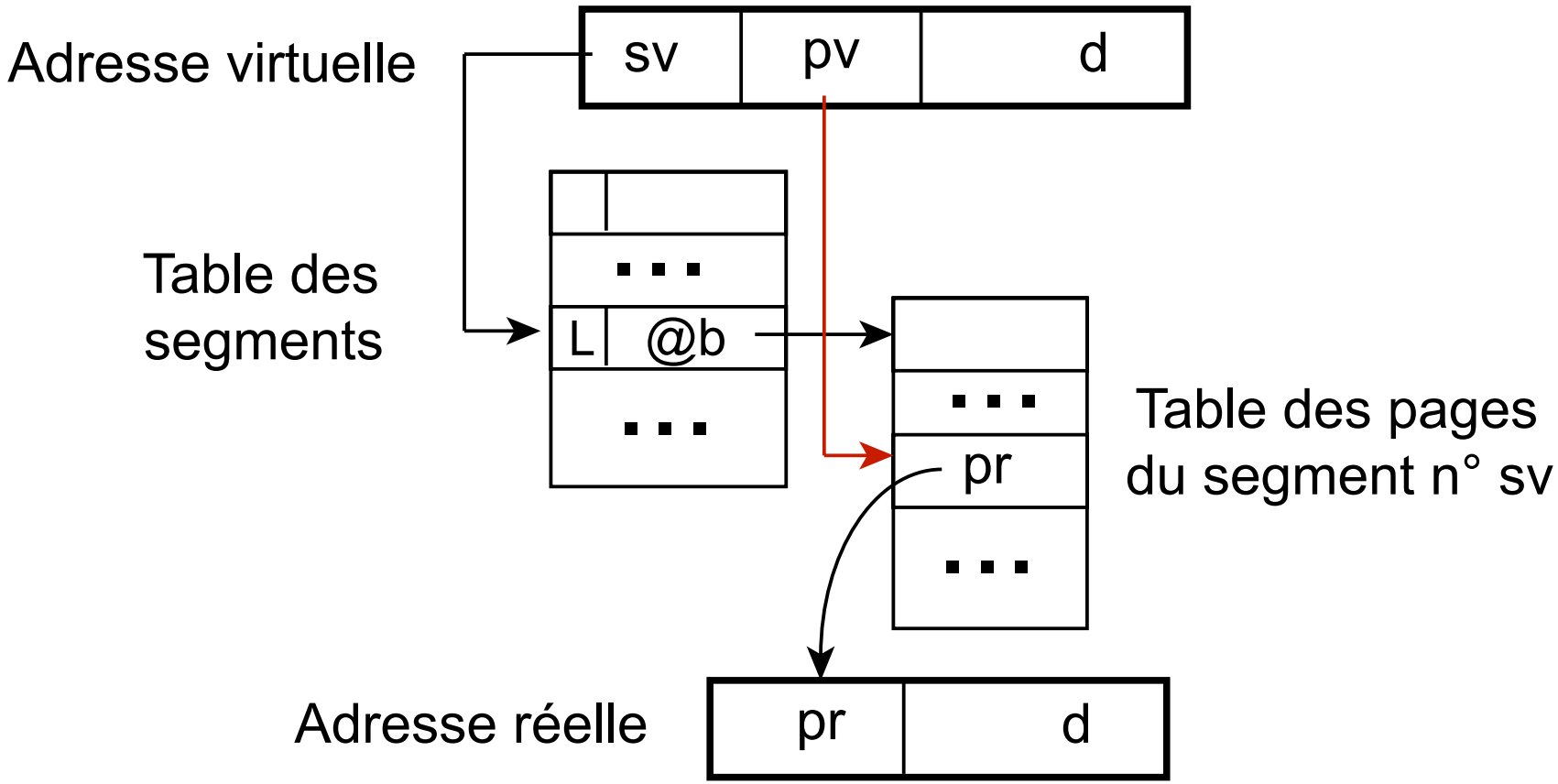
Projection de l'espace virtuel sur l'espace réel

La segmentation-pagination

- Le type segment : typedef $Segment = Page[2^s]$, $s = 58$;
- Structure logique mém. virtuelle : typedef $Mv = Segment[2^n]$;
- Structure logique mémoire réelle : typedef $Mr = Page[2^m]$;



L'adressage segmenté paginé



La segmentation-pagination

Idée de base

Un segment est découpé en bloc de taille fixe, càd en pages ;

Avantages

- Adaptation à la structure logique du programme ;
- Allocation simple des pages réelles.
- Va-et-vient simple.

Inconvénients

- Conversion plus complexe ;
- Deux niveaux de tables.

MMU : optimisation de la convertion d'adresse

Objectif : accélérer la conversion

- Problème : Plusieurs références mémoires pour retrouver une page ;
- Idée : Exploiter la localité des références ;
- Solution : Utiliser une mémoire associative.

Translation Look-aside Buffer (TLB)

- Mémoire associative contenant les dernières références mémoire sous la forme : de couples (pv, pr)

Le couplage d'une image binaire exécutable

Comment charger un programme en mémoire virtuelle ?

Notion de couplage

- Il faut associer un contenu à chaque segment (page) utilisé(e) ;
- Le contenu initial est dans le binaire exécutable ;
- Il faut charger une image binaire exécutable ;
- Le couplage est l'association d'un contenu à une page virtuelle ;
- Le contenu d'un segment est une section de l'image : code, données, pile ;
- Le contenu d'une page est une page de section de l'image.

Technique d'allocation par préchargement

Avantages

- Allocation simplifiée des pages réelles ;
- Pas d'allocation dynamique durant l'exécution ;
- Pas de va-et-vient à gérer.

Inconvénients

- Allocation peu optimale ;
- Impossibilité d'exécuter un programme plus grand que la mémoire réelle.

La pagination à la demande

Principe

- Application du principe de liaison au plus tard (delay binding time) ;
- Une page réelle n'est allouée à une page virtuelle que lorsque la page virtuelle est référencée pour la $1^{ère}$ fois et provoque
  ☞ un défaut de page
- Allocation dynamique des pages réelles à la demande ;
- Problème : toujours trouver une page réelle
  ☞ risque d'interblocage
- Solution : Préemption de pages réelles allouées à des pages virtuelles.

La pagination à la demande

Le traitement d'un défaut de page

Sur occurrence d'une exception émise par le MMU

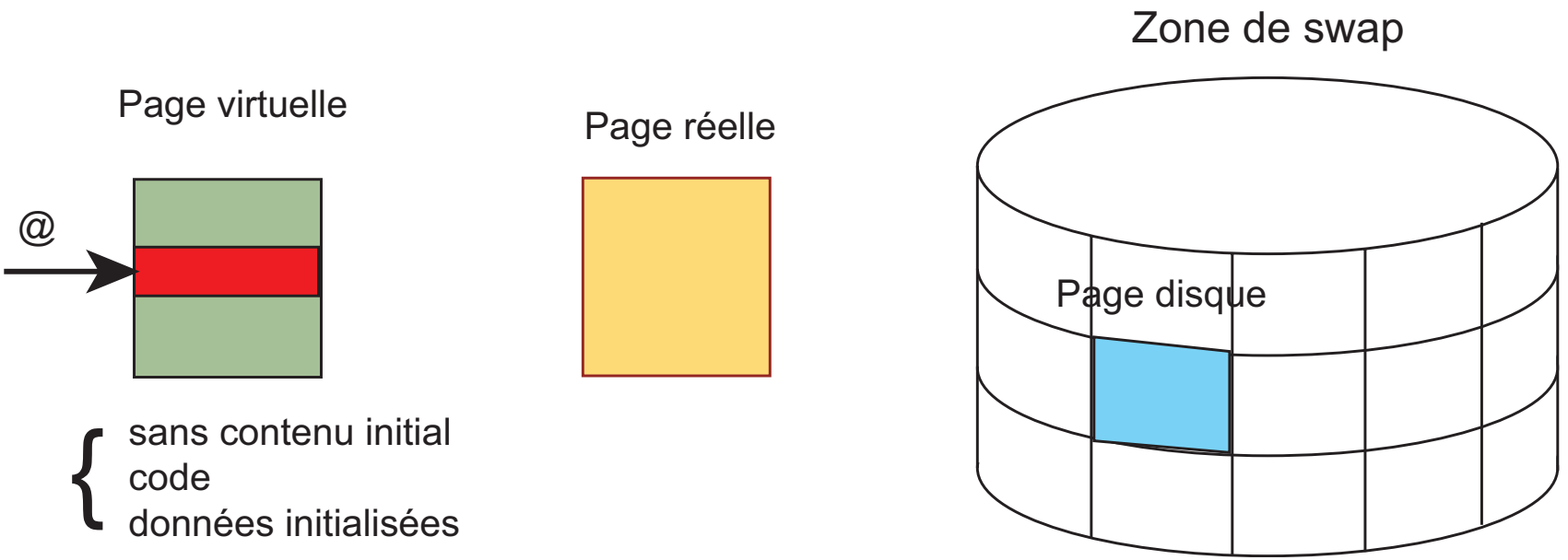
- Allouer une page réelle *pr* ;
- Rendre présente cette page au niveau du MMU ;
- Si nécessaire, charger le contenu couplé à la page virtuelle dans la page réelle *pr* ;
- Retour à l'instruction ayant provoqué le défaut de page.

Problème majeur

- Allouer une page réelle peut nécessiter la préemption d'une page réelle associée à une autre mémoire virtuelle ;
- Objectif : éviter le phénomène d'interblocage ;
- Nécessite de sauvegarder éventuellement le contenu de la page réelle préemptée : ☞ zone de swap.

La pagination à la demande

Le traitement d'un défaut de page



Les stratégies de remplacement de pages

Un problème de fond

- Déterminer les pages virtuelles dont le contenu doit être présent en mémoire ;
- Critère : celles qui vont être référencées dans un futur proche ;
- Problème : on ne connaît pas les références futures ;
- Idée : Regarder le passé en espérant qu'il ressemble au futur.

Critère d'une bonne stratégie

Plus de pages présentes, Moins de défauts de pages

Les stratégies de remplacement de pages

Principales stratégies

- Stratégie RANDOM : choisir la page au hasard ;
- Stratégie FIFO : prendre la page « la plus ancienne » ;
- Stratégie LRU (Least Recently Used) : prendre la page oubliée (non référencée) depuis le plus longtemps ;
- Stratégie NRU (Not Recently Used) : prendre une page oubliée pendant un temps limité.

Les stratégies de remplacement de pages
Stratégie FIFO

Des défauts et peu d'avantages

- prendre la page présente « la plus ancienne » ;
- avantage : stratégie simple ;
- ne vérifie pas le critère de « bonne stratégie » ;
- des pages anciennes ne sont pas là par hasard . . .

☞ pages de segments partagés (bibliothèques dynamiques)

Les stratégies de remplacement de pages

Stratégie LRU

Performante mais difficile à mettre en œuvre

- Idée : choisir la page non référencée depuis le + longtemps ;
- Problème : il faut dater la dernière référence à chaque page ;
- Trop couteux.

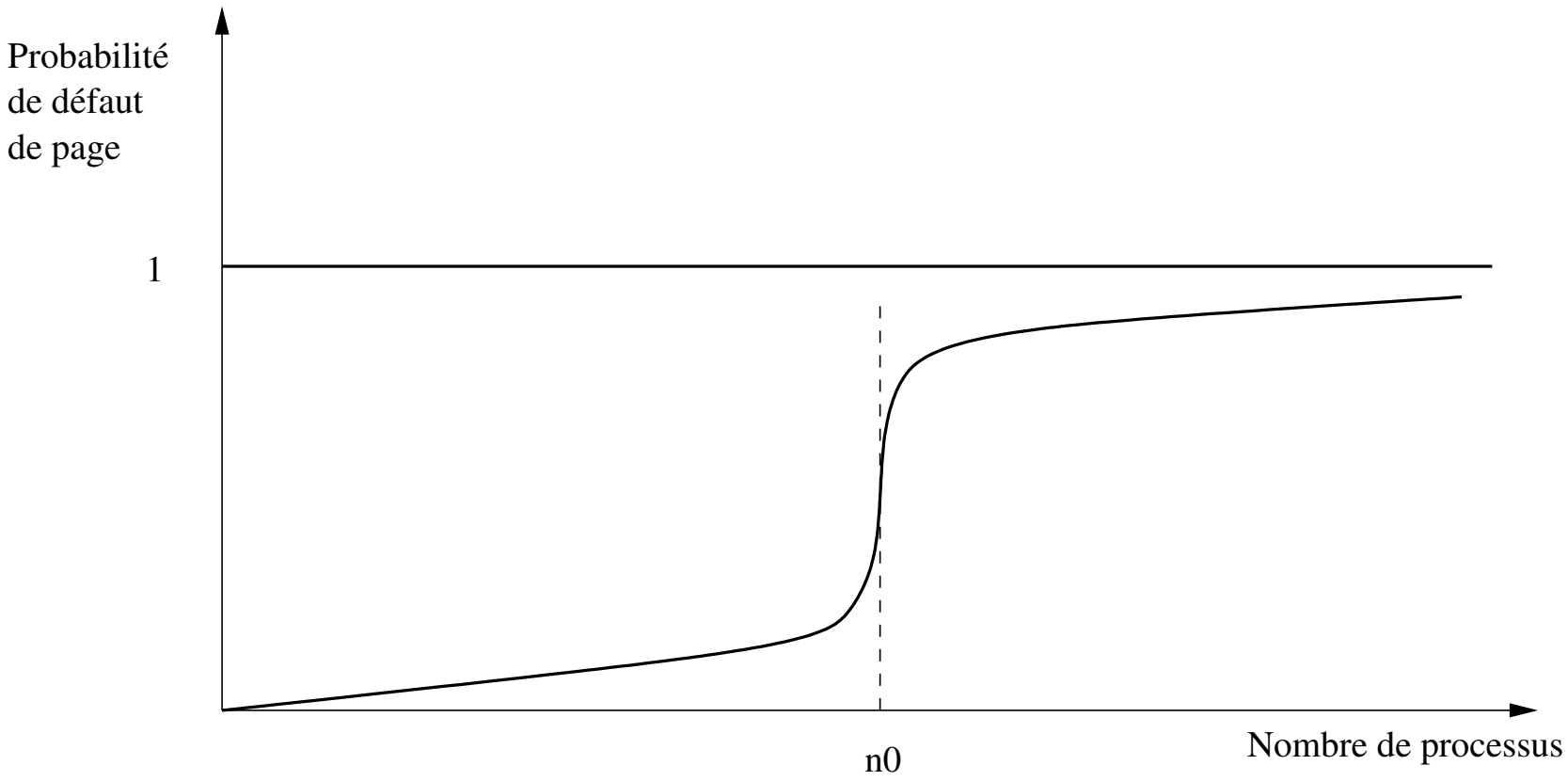
Les stratégies de remplacement de pages

Stratégie NRU

Stratégie LRU approchée

- Choisir parmi les pages non référencées dans un passé récent ;
- Mise en œuvre : utilisation d'un indicateur de référence périodiquement effacé ;
- Existence de 4 états possibles pour une page réelle :
  - non référencée, non écrite ;
  - non référencée, écrite ;
  - référencée, non écrite ;
  - référencée, écrite.

Phénomène d'écroulement



La notion d'espace de travail (*working set*)

Le working set ≡ Espace vital pour un processus

- Lutte contre l'écroulement : donner à chaque processus un nombre suffisant de pages virtuelles présentes ;
- Problème : comment déterminer ce « nombre suffisant » ?
- Principe : capter le taux de défauts de pages des processus ;

Le calcul du working set

- Compter le nombre de défauts de pages sur une période :
  - Si le taux de défauts de page augmente
    $\Rightarrow$ augmenter la taille du working set du processus ;
  - Si le taux de défauts de page diminue
    $\Rightarrow$ diminuer la taille du working set du processus.

La notion d'espace de travail (*working set*)

Méthode calcul

Définition

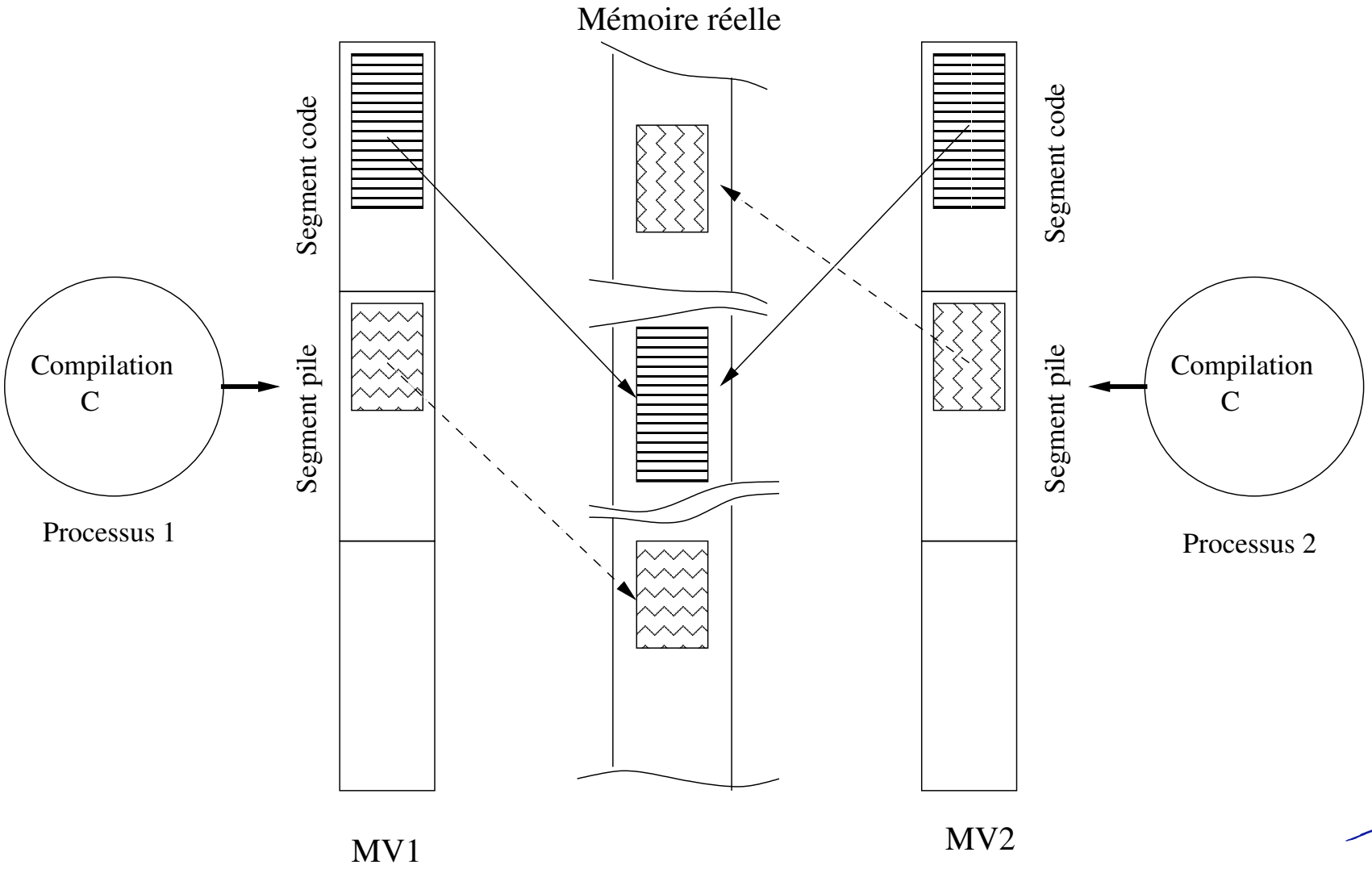
- L'espace de travail à l'instant t est l'ensemble des pages référencées entre les instants $t - \Delta t$) et t ;
- On note $W(t, \Delta t)$, le nombre de pages de l'espace de travail à cet instant t ;
- $W(t, \Delta t)$ évolue (diminue ou augmente) au cours de l'exécution selon le profil des références.

Régulation de charge

- Objectif : maintenir l'invariant : $\sum_1^n W_i(t, \Delta t) \leq M \ \forall t = k\Delta$
- M : nombre de pages réelles disponibles initialement ;
- Eventuellement suspendre des processus si l'invariant est violé.

Optimisations : possibilité de partage

Partage de code



Système Solaris (segmentation + pagination)