Synchronisation des processus

Le problème de l'interblocage

Gérard Padiou

Département Informatique et Mathématiques appliquées
ENSEEIHT

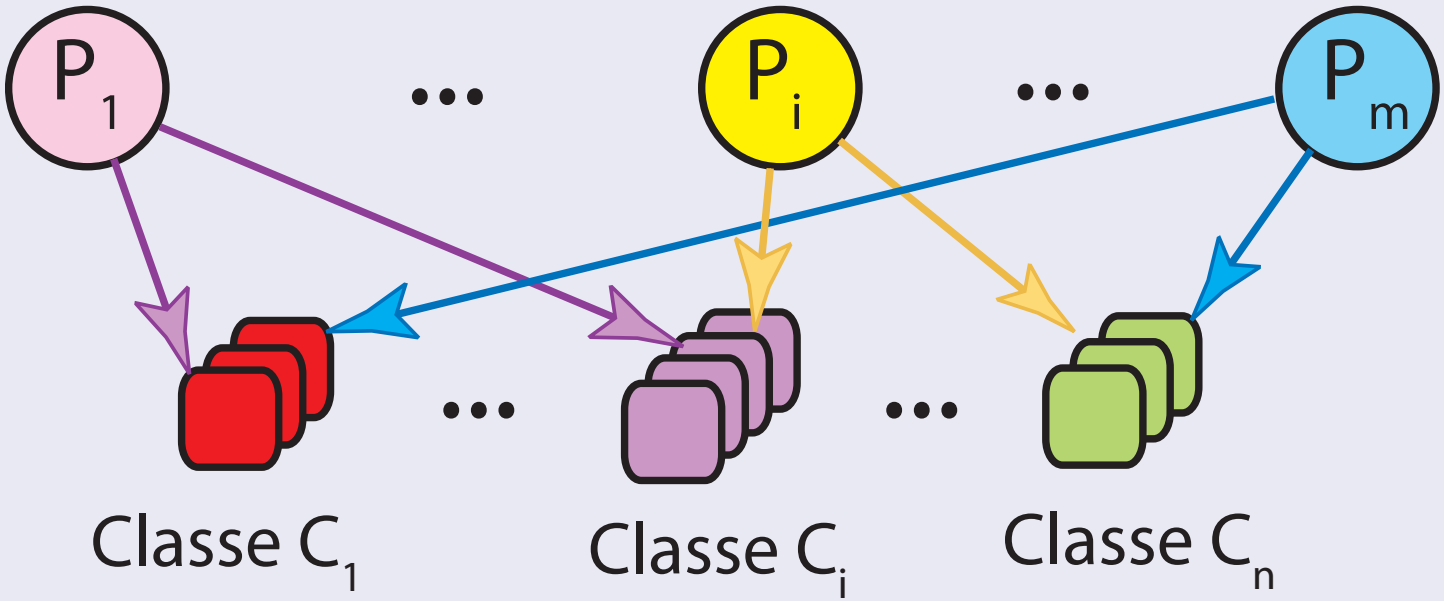
Septembre 2007

plan

Activités parallèles et ressources critiques

Concurrence pour l'accès aux ressources critiques

Tout système composé de processus parallèles accédant à des ressources critiques peut comporter un risque d'interblocage.



Risque d'interblocage

Règles comportementales

- Plusieurs demandes séquentielles ;
- Maintien d'une demande qu'on ne peut satisfaire ;
- Conservation des ressources acquises.

Exemple

```
process P1() {
    ...
    demander(A) ;
    demander(B) ;
    ...
    restituer(B) ;
    restituer(A) ;
}
```

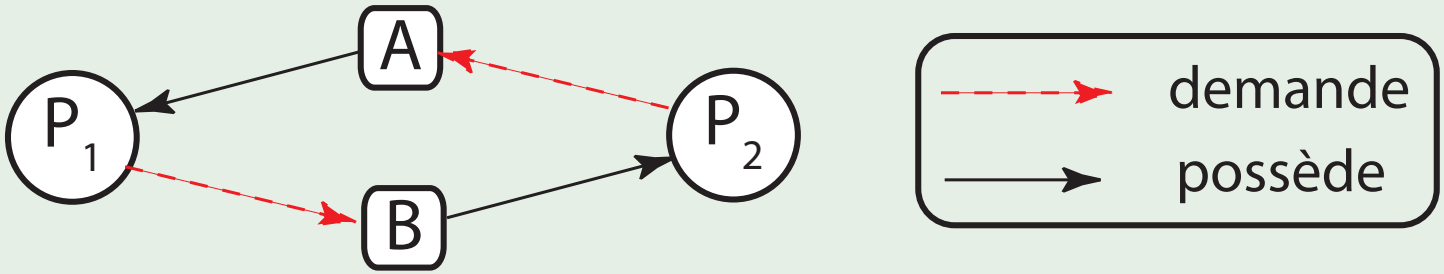
```
process P2() {
    ...
    demander(B) ;
    demander(A) ;
    ...
    restituer(A) ;
    restituer(B) ;
}
```

Interblocage entre deux processus

Abstraction du calcul par un graphe

- Sommet ≡ Processus ou Ressource
- Arc : Processus → Ressource si le processus est bloqué sur sa demande (ressource occupée par un autre processus) ;
- Arc : Ressource → Processus si le processus a la ressource.

Exemple

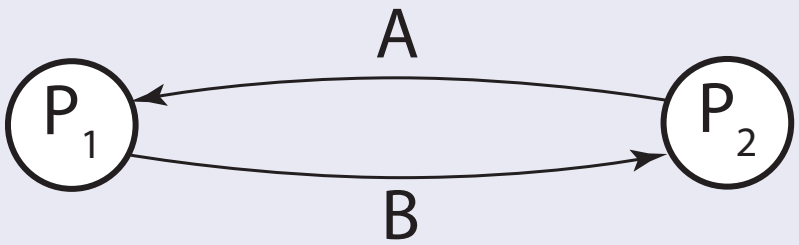


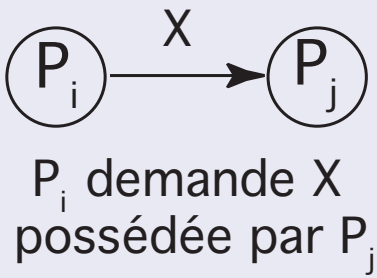
Variante de graphe

Graphe d'attente de ressource (wait-for graph)

Simplification possible lorsqu'un processus demande une ressource spécifique identifiée :

- Sommet ≡ processus ;
- Arc étiqueté par la ressource demandée.





☞ Cas fréquent dans les systèmes de bases de données

Risque d'interblocage

Conditions nécessaires

1. **Criticité** des ressources ;
2. **Pas d'abandon** : un processus bloqué sur une demande la maintient et garde les ressources qu'il possède déjà.
3. **Pas de réquisition** : un processus bloqué sur une demande ne perd pas les ressources qu'il possède déjà : pas de réquisition des ressources possédées par un processus bloqué demandeur pour les donner à un autre ;
4. **Cycle** présent dans le graphe d'allocation.

Techniques de prévention

Idée

Il suffit de rendre une des conditions nécessaires fausse durant toute exécution du système (invariant).

Granularité d'observation : pas d'allocation/libération en cours.

Exemple : Prévention des cycles (condition 4)

- en ordonnant totalement les ressources (classes de)[a] ;
- Choix de l'ordre important : impose un ordre d'acquisition empéchant un usage optimal des ressources (ressource demandée trop tôt).

[a] James W. Havender, Avoiding Deadlock in Multitasking Systems, IBM Systems Journal, 7(2) :74-84, 1968.

Techniques de prévention (suite)

Eliminer la ressource critique (condition 1) ?

Notion de ressource virtuelle

- Ressource critique $\equiv$ ressource en nombre limité (une seule) ;
- Ressource virtuelle : à la demande et même fonctionnalité ;
- Passage d'une ressource commune mais critique à des ressources privées (chaque processus à sa ressource virtuelle)
  $\Rightarrow$ évite toute concurrence d'accès.

Exemple

- Imprimante virtuelle implantée par un fichier ;
- Gestion d'une liste des fichiers à imprimer ;
- Un processus dédié (*spool*) imprime les fichiers
  ($\Rightarrow$ imprimante réelle allouée statiquement).

Techniques de prévention (suite)

Restitution des ressources obtenues (condition 2) ?

Idée de phase d'acquisition qui réussit ou échoue

- Si une demande ne peut être satisfaite immédiatement, le processus libère les ressources qu'il avait obtenues ;
- Si échec durant la phase d'acquisition, il faut recommencer ;
- Problème : risque de famine ;
- Usage : bases de données, systèmes transactionnels.

Solution

Idée : Définir un ordre total sur les processus demandeurs

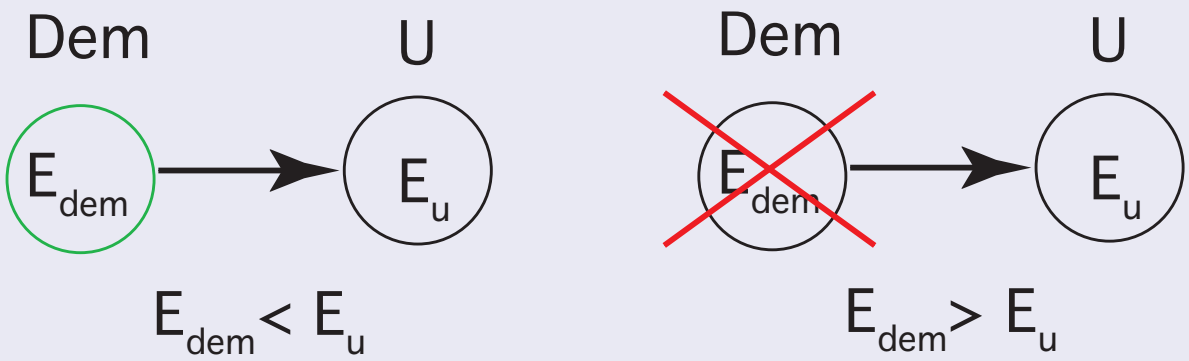
- Notion d'estampille : numéro du processus, date de création,. . . ;
- Le plus « ancien » finira par réussir sa phase d'acquisition.

Techniques de prévention (suite)

Algorithme *wait/wound/die* (Rosenkrantz 1978)

Version wait/die

- if ($E_{dem} < E_u$) recommencer (die) ;
- if ($E_{dem} > E_u$) attendre (wait).



☞ Pas de cycle d'attente : $E_{dem_1} < E_{dem_2} < \ldots < E_u$

Techniques de prévention (suite)

Réquisition des ressources obtenues (condition 3) ?

Principe

- Réquisition d'une ressource déjà occupée ;
- Nécessite de sauvegarder l'état courant de la ressource avant de la donner à un autre processus ;
- Risque de surchauffe !

Utilisation : les mémoires virtuelles

- Approche utilisée dans systèmes de pagination à la demande ;
- Une page réelle appartenant à un processus (à sa mémoire virtuelle) peut être réquisitionnée pour un autre ;
- Nécessite de sauvegarder éventuellement son contenu avant son changement de propriétaire. . . ($\Rightarrow$ une zone de swap).

Détection des interblocages

Détection

- Propriété stable ;
- Gestion du graphe d'allocation ;
- Détection $\equiv$ cycle dans le graphe (condition 4).

Guérison

- Casser le cycle $\Rightarrow$ détruire un processus ;
- Problème : choix de la victime ($+$récent ? $-$urgent ? . . .) ;
- ☞ Prévoir des points de reprise.