INTERBLOCAGE

Plan

- Modèle de Système
- Caractérisation de l'interblocage
- Approches du problème
- Formalisation de l'interblocage
- Une approche combinée de différentes méthodes

Modèle de Système

- Environnement multiprogrammé
- Nombre fini de ressources partagées par des processus concurrents

Modèle de Système

- **Ressource**
  Tout objet informatique matériel ou logiciel nécessaire à l'exécution d'un programme

Modèle de Système

- **Caractérisation des ressources**
  - matérielle ou logicielle
  - à accès unique ou multiple
  - banalisée ou unique
  - virtuelle ou physique
  - préemptible ou non préemptible

Modèle de Système

Caractérisation des ressources

Ressources banalisées ou unique

- Ressource banalisée : ensemble d'instances de ressource identiques et allouables indifféremment, appelée **classe**

Exemple : les pages d'une mémoire paginée

- Ressource unique : 1 seul exemplaire de la ressource

Modèle de Système

Caractérisation des ressources

Ressource préemptible ou non préemptible

- Préemptible : la ressource peut être retirée à un processus sans dommage

Exemple : pages de mémoire dans une gestion de mémoire paginée

- Non préemptible : la ressource ne peut être retirée

Exemple : une imprimante

Modèle de Système

Utilisation d'une ressource

On considère trois étapes :

1. Demande de la ressource : attente du processus si la ressource est détenue par un autre processus
2. Utilisation de la ressource
3. Libération de la ressource

Exemple introductif

Allocation dynamique de mémoire

Réserve = 16 pages allouables à 3 processus

P1	P2	P3
demande 3 pages	demande 4 pages reste 5 pages	demande 4 pages
demande 2 pages	demande 1 page reste 0 page	demande 2 pages
demande 1 page	demande 2 pages	demande 2 pages

Interblocage : tous les processus sont bloqués

Exemple introductif

Accès en écriture à des fichiers

2 fichiers allouables à 2 processus

P1	P2
ouvrir(F, écriture)	ouvrir(G, écriture)
ouvrir(G, écriture)	ouvrir(F, écriture)
…	…
fermer(F)	fermer(F)
fermer(G)	fermer(G)

Interblocage : si on alloue F à P1, puis G à P2, les 2 processus sont bloqués

Caractérisation de l'interblocage

Définition

Un ensemble de processus est en interblocage si chacun est en attente d'un événement qui ne peut être produit que par un processus de l'ensemble.

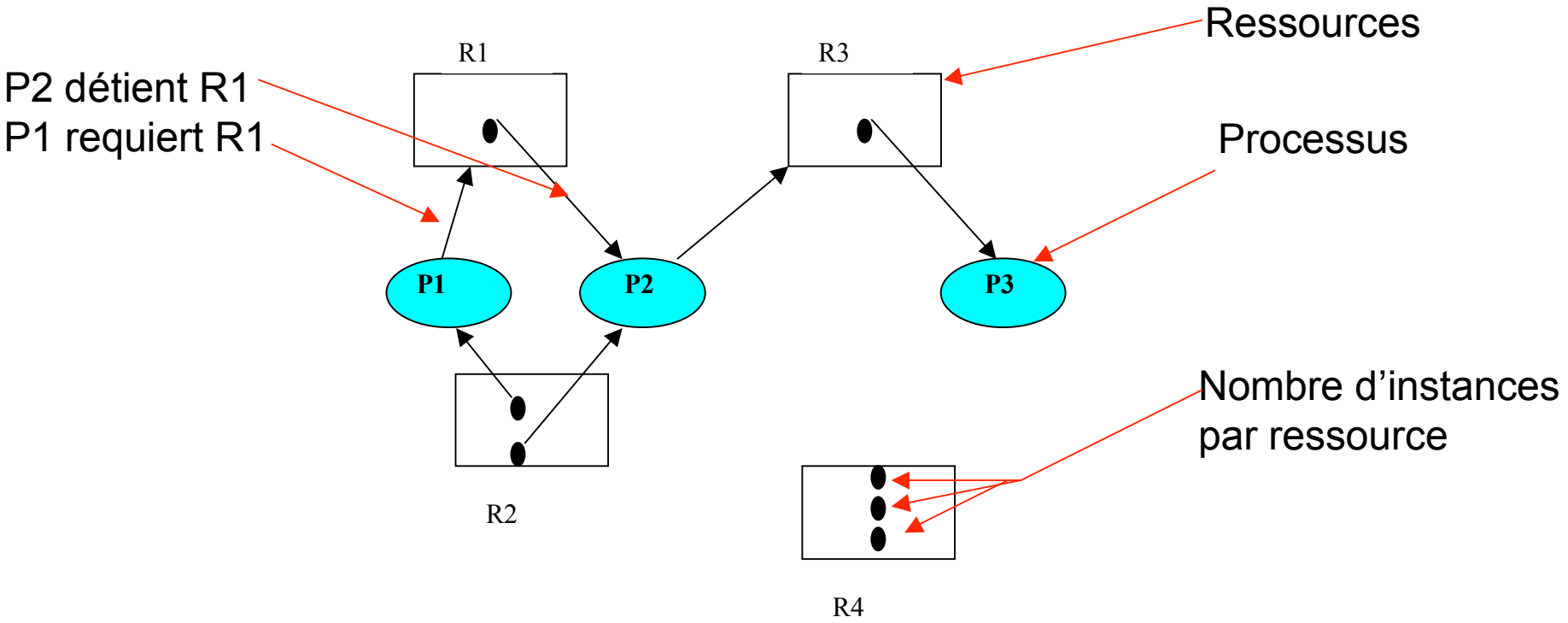
Le plus souvent l'événement attendu est la libération d'une ressource par un processus de l'ensemble

Caractérisation de l'interblocage

Conditions nécessaires

1. Exclusion mutuelle : ressource allouée exclusivement à un processus
2. Allocation dynamique et accumulation des ressources
3. Pas de préemption des ressources acquises
4. Attente circulaire : il existe un cycle de processus : P_0 attend P_1 attend P_2….P_n attend P_0

Modélisation (Holt)
Graphe d'allocation de ressources



Gestion de l'interblocage

Approches

1. Ignorer le problème
2. Détection puis guérison
3. Prévention en empêchant l'apparition d'une condition nécessaire
4. Évitement en allouant les ressources avec précaution

1. La politique de l'autruche

La plupart des systèmes ignorent le problème

UNIX, Windows

- on suppose que la survenue d'un interblocage est peu probable
- Le coût de la prévention ou de la détection en termes de contraintes pour l'utilisateur et de ressources pour le système est élevé

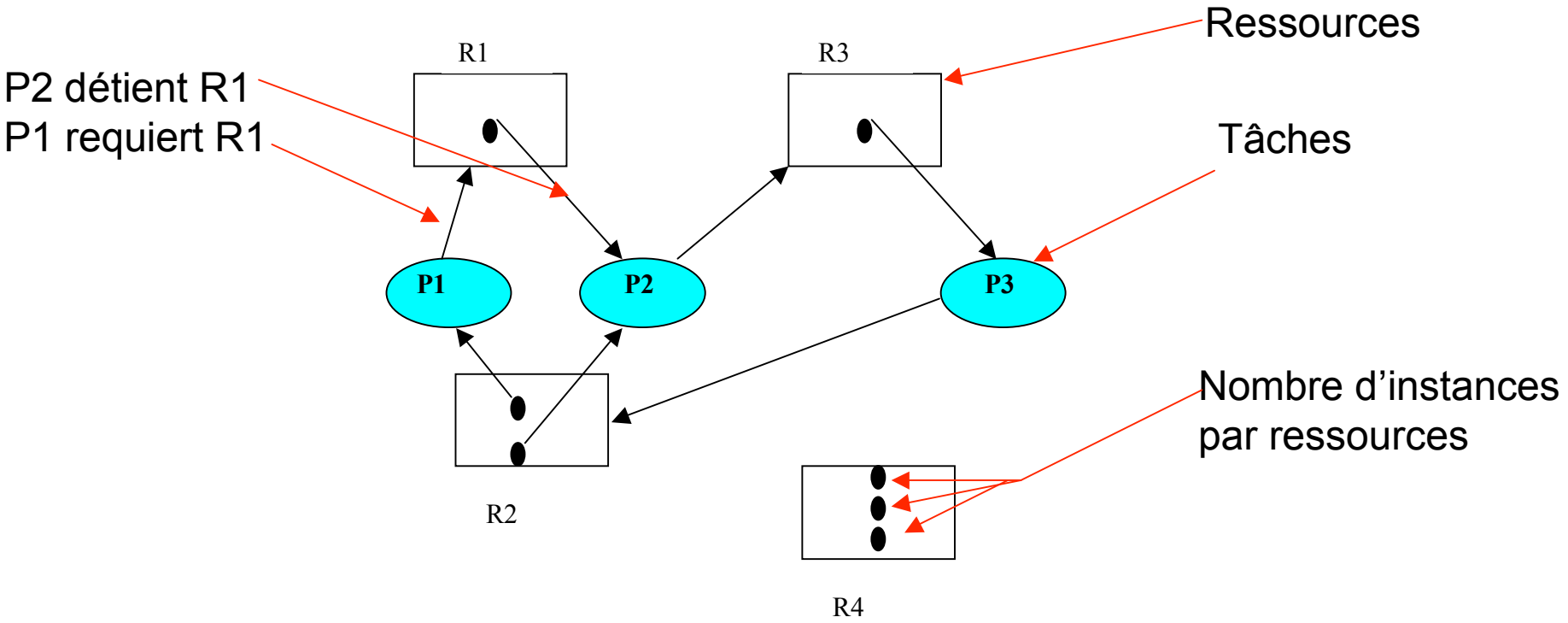
2. Détection Guérison

Détection

Cas 1 Une seule instance par ressource
interblocage s'il existe un cycle dans le graphe

Cas 2 Plusieurs instances par ressource
L'existence d'un cycle n'implique pas forcément un interblocage; la non-existence de cycle implique qu'il n'y a pas interblocage

Exemple d'interblocage



Détection d'un interblocage

Cas d'une instance par ressource

Utiliser un algorithme de recherche d'un cycle dans un graphe complexité en $0(n^2)$ où n est le nombre d'arcs.
En général, nombreux processus et ressources
Algorithme coûteux
Inadapté si plusieurs instances de ressources

Guérison d'un interblocage

- **Préemption de ressource**

Retirer temporairement une ressource à un processus pour l'attribuer à un autre.
Dépend du type de ressource, demande souvent une intervention manuelle

- **Destruction de processus**

Choix du ou des processus?

Guérison d'un interblocage

- **Reprise arrière ou rollback après préemption**

Principe : périodiquement enregistrer l'état de chaque processus

Restaurer l'état du processus avant l'acquisition de la ressource par le processus et sa préemption

Le travail depuis le dernier point de reprise est perdu

Guérison d'un interblocage

- **Reprise arrière ou rollback après préemption**

Les questions :

1 Le choix d'une victime

2 Reprise totale

3 Risque de famine

2. Prévention des interblocages

Supprimer une des conditions nécessaires à la survenue d'un interblocage :

1. La condition d'exclusion mutuelle

En général impossible exemples partage d'une imprimante, d'une page mémoire.

Pour certains périphériques, mécanisme de spoule permettant de différer l'utilisation de la ressource

Idée : n'attribuer une ressource que si absolument nécessaire

2. Allocation dynamique et accumulation

Les processus demandent toutes les ressources nécessaires à leur exécution.

Allocation globale

Inconvénients :

- un processus doit connaître tous ses besoins en ressources
- Mauvaise utilisation des ressources
- Famine possible de certains processus

Applicable dans les systèmes à traitement par lots

2. Allocation dynamique et accumulation

Allocation par classes ordonnées

Classe =ensemble des instances d'une ressource

Les classes sont ordonnées a priori

- demande et allocation dans l'ordre
- demande globale par classe

Meilleure utilisation des ressources si l'ordre est bien choisi.

OS/MVT fichiers, mémoire, périphériques

2. Allocation dynamique et accumulation allocation par classes ordonnées

2 processus : P1 P2, 2 ressources : R1 R2

Soit la séquence suivante :

P1 demande et obtient R1, P2 demande et obtient R2

P1 demande R2 : attente, P2 demande R1 : attente

interblocage

Si R1 et R2 sont demandées dans le même ordre,

Un seul des 2 processus attend, pas d'interblocage

3. Condition de non-préemption

difficile, voire impossible

4. Condition d'attente circulaire

Utiliser la méthode des classes ordonnées :

- on évite l'attente circulaire
- souvent difficile de trouver un ordre satisfaisant nombre de ressources élevé et diversité de leur utilisation

Résumé des approches de prévention statique

Condition	Approche
Exclusion mutuelle	Traitement en différé
Allocation dynamique et accumulation	Allocation globale
Non-préemption	Retirer des ressources
Attente circulaire	Allocation par classes ordonnées

4. Prévention dynamique

Chaque processus annonce son besoin maximum en ressources.

Rôle de l'allocateur :

- vérifier avant de satisfaire une requête, qu'il pourra alors satisfaire toutes les requêtes futures, l'état du système est dit **fiable**
- dans le cas contraire, retarder l'allocation en bloquant momentanément le processus, car il y a risque d'interblocage.

Formalisation de l'interblocage

Système = (P, E, X, A, R)

P={P_1, P_2,…P_n} ensemble fini de processus

E={R_1, R_2,…R} ensemble de classes de ressources

X= (x_i) x_i nombre de ressources initial de la classe R_i

A(t)=(a_{ij}) a_{ij} nombre de ressources de la classe R_j allouées au processus P_i à l'instant t

R(t)=(r_{ij}) r_{ij} nouvelle requête de ressources de la classe R_j par le processus P_i à l'instant t

Formalisation de l'interblocage

Conventions

- Pour alléger la notation, on ne mentionnera pas le moment de l'observation t : A(t) est noté A

A[i] représente le vecteur courant d'allocation de ressources pour le processus P_i

$$A = \sum_{i=1..m} A[i]$$

Idem pour les autres matrices

- Soient X et Y vecteurs de taille n

$X \leq Y$ si $X[i] \leq Y[i]$ tout $i=1 \ldots n$

Formalisation de l'interblocage

Changements d'états du système

Soit **D=X-A** la disponibilité en ressources

- Requête P_i demande à l'instant t une nouvelle requête N

R[i]<– R[i] + N

- Acquisition allocation à P_i de M

A[i]<–A[i] + M en fait M=R[i] et R[i]<– 0

D<– D - M

- Libération par P_i de H

A[i]<– A[i] - H

D<–D + H

Formalisation de l'interblocage

Hypothèses

• tout processus libère au bout d'un temps fini ses ressources

• l'état du système est **réalisable** :

- la demande d'un processus ne peut excéder le nombre initial de ressources

- un processus ne peut acquérir plus de ressources qu'il n'a demandé

- la somme des allocations aux processus ne peut dépasser le nombre initial de ressources

Détection de l'interblocage

Absence d'interblocage

Il existe une suite d'états réalisables telle que tout processus obtienne sa requête à l'instant considéré et s'exécute jusqu'à sa fin.

Action de l'allocateur

Ranger les processus dans un ordre tel que leur requête soit satisfaite s'ils s'exécutent dans cet ordre.

Interblocage si on ne peut inclure au moins un processus dans la suite de processus.

Détection de l'interblocage

État sain

Suite saine

Soient S = suite de processus

S(k) le rang du processus dans la suite

S est une suite saine si à l'instant considéré, on vérifie

tout P_i , $R[i] \leq D + \sum_{S(j)<S(i)} A[j]$

État sain

Il existe une suite saine contenant tous les processus

Absence d'interblocage = état sain

Détection de l'interblocage

Algorithme de détection de l'interblocage

Soient : T[1..m] les ressources potentiellement disponibles

F[1..n] F[i]= vrai si P_i est dans la suite saine, faux sinon

1. initialisation T=D ressources disponibles, trouvé=vrai; F[i]=faux pour tout processus

2. tant que trouvé et il existe P_j t.q. non F[j] faire

    rechercher i tel que non F[i] et R[i]≤T;

    si i n'existe pas alors trouvé=faux;

    sinon F[i]=vrai; T=T+A[i];finsi;fait;

3. si pour tout i, F[i]==vrai, alors pas d'interblocage sinon les processus t.q. F[i]==faux sont en interblocage

Exemple

m=3 n=5 X=(7,2,6)

	Allocation A B C	Requête A B C	Disponible A B C
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 0	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

Exemple(suite)

F=(faux,faux,faux,faux,faux) T=(0,0,0)
Choix de P_0 : F=(vrai,faux,faux,faux,faux) T=(0,1,0)
Choix de P_2 : F=(vrai,faux,vrai,faux,faux) T=(3,1,3)
Choix de P_3 : F=(vrai,faux,vrai,vrai,faux) T=(5, 2,4)
Choix de P_1 : F=(vrai,vrai,vrai,vrai,faux) T=(7, 2,4)
Choix de P_4 : F=(vrai,vrai,vrai,vrai,vrai) T=(7, 2,6)

(P_0, P_2, P_3, P_1, P_4) est une suite saine complète
Il n'y a pas d'interblocage.

Exemple(suite)

Supposons que P_2 demande une ressource de type C

	Allocation	Requête	Disponible
	A B C	A B C	A B C
P_0	0 1 0	0 0 0	0 0 0
P_1	2 0 0	2 0 2	
P_2	3 0 3	0 0 1	
P_3	2 1 1	1 0 0	
P_4	0 0 2	0 0 2	

Choix de P_0 : T=(0 1 0) : on ne peut satisfaire aucun autre processus.

P_1, P_2, P3, P_4 sont en interblocage

Détection Conclusion

Complexité de l'algorithme

en temps $O(m \times n^2)$

en mémoire $O(m \times n)$

L'exécution de l'algorithme à chaque nouvelle requête est prohibitive en temps

Nombre fixe de ressources et de processus

Reconfigurer les matrices en cas de changements

Prévention dynamique

Un processus doit annoncer son besoin maximum en ressources

Max=[$\mathbf{max_{ij}}$] max_{ij} besoin maximum du processus P_i pour la ressource R_j

B=[$\mathbf{b_{ij}}$] $b_{ij}=max_{ij} - a_{ij}$ besoin de P_i pour atteindre son maximum à l'instant considéré.

Prévention dynamique

Pas de risque d'interblocage

Il existe une suite d'états sains t.q. tout processus peut obtenir son besoin maximum et s'exécuter jusqu'à sa fin : **état fiable**

Hypothèses

tout P_i , $Max[i] \leq X$, $A[i] \leq Max[i]$, $A[i] + B[i] \leq Max[i]$

Suite fiable

S=suite de processus

tout P_i dans S, $B[i] \leq D + \sum_{S(j)<S(i)} A[j]$

Si S contient tous les processus le système est dans un **état fiable**

Prévention dynamique

Action de l'allocateur

-Ranger les processus dans un ordre tel que leur besoin maximum puisse être satisfait s'ils s'exécutent dans cet ordre état fiable.

Il y a risque d'interblocage, si on ne peut inclure au moins un processus dans la suite de processus.

-Satisfaire une requête si partant d'un état fiable, cette allocation laisse le système dans un état fiable sinon différer l'allocation.

Prévention dynamique

Algorithme du banquier (Dijkstra)

- détection d'un état fiable

Soient : T[1..m] les ressources potentiellement disponibles

F[1..n] F[i]= vrai si P_i est dans la suite fiable, faux sinon

1. initialisation T=D ressources disponibles, trouvé=vrai;

F[i]=faux pour tout processus

2. tant que trouvé et il existe P_j t.q.F[j]==faux faire

    rechercher i tel que non F[i] et B[i]≤T;

    si i n'existe pas alors trouvé=faux;

    sinon F[i]=vrai; T=T+A[i];finsi;fait;

3. si pour tout i, F[i]==vrai, alors pas d'interblocage

l'état du système est fiable.

Prévention dynamique

Algorithme du banquier (Dijkstra)

- Partant d'un état fiable, satisfaire une requête d'un processus P_i, R[i] si l'état reste fiable.

Simuler l'allocation :

```
D = D - R[i];
A[i] = A[i] + R[i];
B[i] = B[i] - R[i];
```

Appliquer l'algorithme de l'état fiable :
si l'état reste fiable, alors satisfaire la requête
sinon différer l'allocation.

Prévention dynamique

Exemple

m = 3, n=5, X = (10,5,7)

À un instant t, on a l'état suivant :

	Allocation	Max	Besoin	Disponible
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	7 4 3	3 3 2
P_1	2 0 0	3 2 2	1 2 2	
P_2	3 0 2	9 0 2	6 0 0	
P_3	2 1 1	2 2 2	0 1 1	
P_4	0 0 2	4 3 3	4 3 1	

Prévention dynamique

Exemple (suite)

Choix de P_1: F=(faux,vrai,faux,faux,faux) T=(5,3,2)
Choix de P_3: F=(faux,vrai,faux,vrai,faux) T=(7,4,3)
Choix de P_4: F=(faux,vrai,faux,vrai,vrai) T=(7,4,5)
Choix de P_2: F=(faux,vrai,vrai,vrai,vrai) T=(10,4,7)
Choix de P_0: F=(vrai,vrai,vrai,vrai,vrai) T=(10,5,7)
L'état est fiable

Prévention dynamique

Exemple (suite)

Supposons que R[1]=(1,0,2)

On simule l'allocation, l'état est fiable, on satisfait la requête de P_1

	Allocation	Max	Besoin	Disponible
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	7 4 3	2 3 0
P_1	3 0 2	3 2 2	0 2 0	
P_2	3 0 2	9 0 2	6 0 0	
P_3	2 1 1	2 2 2	0 1 1	
P_4	0 0 2	4 3 3	4 3 1	

Prévention dynamique

Exemple (suite)

Supposons que R[4]=(3,3,0) pas assez de ressources disponibles, la requête de P_4 est différée

	Allocation	Max	Besoin	Disponible
	A B C	A B C	A B C	A B C
P_0	0 1 0	7 5 3	7 4 3	2 3 0
P_1	3 0 2	3 2 2	0 2 0	
P_2	3 0 2	9 0 2	6 0 0	
P_3	2 1 1	2 2 2	0 1 1	
P_4	3 3 2	4 3 3	4 3 1	

Prévention dynamique

Exemple (suite)

Supposons que R[0]=(0,2,0)

Le besoin d'aucun processus ne peut être satisfait

État non fiable l'allocation est différée

	Allocation	Max	Besoin	Disponible
	A B C	A B C	A B C	A B C
P_0	2 3 0	7 5 3	5 2 3	2 1 0
P_1	3 0 2	3 2 2	0 2 0	
P_2	3 0 2	9 0 2	6 0 0	
P_3	2 1 1	2 2 2	0 1 1	
P_4	3 3 2	4 3 3	4 3 1	

Prévention dynamique

Complexité de l'algorithme

en temps $O(m \times n^2)$

en mémoire $O(m \times n)$

L'exécution de l'algorithme à chaque requête est prohibitive en temps.

Contrainte pour les processus

Annoncer leur besoin maximum

Nombre fixe de ressources et de processus

Reconfigurer les matrices en cas de changements

Prévention dynamique

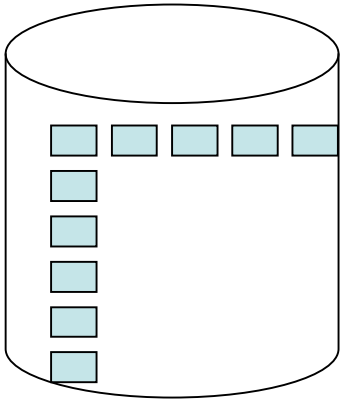
Cas d'une seule classe de ressource

- Peut être appliqué plus aisément par exemple pour des blocs disques, des pages de mémoire
- Simplification des algorithmes :

Trier les processus par besoin max croissant (B[i]), si on peut satisfaire le processus candidat, puis le premier processus de la suite triée alors succès, sinon échec.

S'il y a égalité des Max[i], alors il suffit de ne faire une allocation que si on peut satisfaire le processus le mieux pourvu.

Exemple



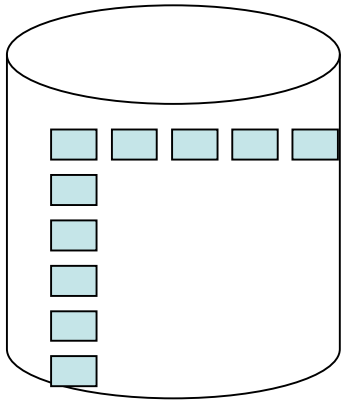
50 clusters disques

Disponible = 50 – (14+7+9+14)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	**3**	**2**
3	9	**4**	**1**
6	5	7	**2**
7	5	2	**2**
1		2	9
		1	
TOTAUX 22	26	21	23

t1	**Allocation**	**Requête**	**Disponible**
P1	14	7	6
P2	7	9	
P3	9	7	
P4	14	9	

Aucune requête ne peut être satisfaite avec le disponible restant **Interblocage**



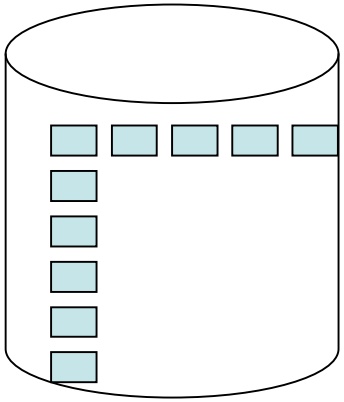
50 clusters disques

Disponible = 50 – (14+21+2+12)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	3	**2**
3	**9**	4	**1**
6	**5**	7	**2**
7	5	2	2
1		2	9
		1	
TOTAUX 22	26	21	23

t2	**Allocation**	**Requête**	**Disponible**
P1	14	7	1
P2	21	5	
P3	2	3	
P4	12	2	

Aucune requête ne peut être satisfaite avec le disponible restant **Interblocage**



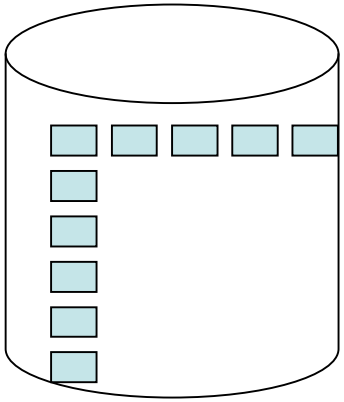
50 clusters disques

Disponible = 50 – (8+16+9+14)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	**3**	**2**
3	**9**	**4**	**1**
6	5	7	**2**
7	5	2	**2**
1		2	9
		1	
TOTAUX 22	26	21	23

t3	**Allocation**	**Requête**	**Disponible**
P1	8	6	3
P2	16	5	
P3	9	7	
P4	14	9	

Aucune requête ne peut être satisfaite avec le disponible restant **Interblocage**



50 clusters disques

Disponible = 50 – (14+7+2+9)

P1	P2	P3	P4
1	**4**	**2**	**7**
4	**3**	3	**2**
3	9	4	1
6	5	7	2
7	5	2	2
1		2	9
		1	
22	26	21	23

t4	**Allocation**	**Requête**	**Besoin max**	**Disponible**
P1	14	7	12	18
P2	7	9	19	
P3	2	3	24	
P4	9	1	17	

On suppose MAX = 26 tout P_i

État sans interblocage

- Servir Pi tel que le disponible restant permette d'atteindre le maximum pour une des tâches restantes

  Servir P1 (requete = 7) (disponible = 11)
  (besoin= 5) P1 peut atteindre son Max=26
  Servir P2 (requete=9) (disponible=9) (besoin=10)
  Aucun processus ne peut atteindre 26
  Servir P3 (requete=3) (disponible=15)
  (besoin=21) aucun processus ne peut atteindre 26
  Servir P4 (requete=1) (disponible=17)
  (besoin=16) P1 peut atteindre 26
  On peut servir P1 ou P4 sans risque d'interblocage

Gestion de l'interblocage par une approche combinée de différentes méthodes

Principe

- Partitionner les ressources en classes hiérarchiquement ordonnées
- Appliquer la méthode des classes ordonnées
- A l'intérieur de chaque classe, appliquer la technique la plus appropriée prévention statique, dynamique, détection, guérison

Gestion de l'interblocage par une approche combinée de différentes méthodes

Exemple

Système composé de 4 classes de ressources :

- **Ressources internes :** ressources utilisées par le système, par exemple bloc de contrôle d'un processus
- **Mémoire centrale :** mémoire utilisée par un processus
- **Resssources d'un processus :** fichiers, périphériques à accès unique
- **Espace de va-et-vient**

Gestion de l'interblocage par une approche combinée de différentes méthodes

Technique à l'intérieur de chaque classe

Ressources internes : prévention statique méthode des classes ordonnées
Mémoire centrale : prévention par préemption (swapping de processus)
Resssources d'un processus : prévention statique ou dynamique
Espace de va-et-vient : faire une préallocation, on connaît en général la taille maximale de l'espace de va-et-vient pour un processus

Bibliographie

- C.Kaiser *Cours Cnam NFP137 2006*
- B.Lécussan *Cours Cnam NFP137 2007*
- Silberschatz et alt. *Operating System Concepts*

Wiley 2007

- Tanenbaum A. *Systèmes d'Exploitation*

Pearson Education 2003

- S. Bouzefrane *Les systèmes d'exploitation*

Unix, Linux et Windows XP avec C et Java

Dunod 2005